IBM Planning Analytics
Version 2 Release 0

Planning Analytics Workspace

IBM

Note

Before you use this information and the product it supports, read the information in "Notices" on page 651.

Product Information

This document applies to IBM Planning Analytics Version 2.0 and might also apply to subsequent releases.

Licensed Materials - Property of IBM

Last updated: 2020-10-07

Contents

Welcome to Planning Analytics Workspace

IBM® Planning Analytics Workspace is a web-based interface for IBM Planning Analytics. It provides an interface to TM1® data, with ways to plan, create, and analyze your content.

Planning Analytics Workspace helps you focus on the things that matter to your business. Using Planning Analytics Workspace, you can identify and understand patterns and relationships in data. You can use this information to understand why things happen, and predict what might happen. Planning Analytics Workspace opens up the world of advanced analytics to all business users.

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products.

IBM HTML documentation has accessibility features. PDF documents are supplemental and include no extra accessibility features. For accessible documentation, see IBM Knowledge Center.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

The Sample Outdoors Company, Great Outdoors Company, GO Sales, any variation of the Sample Outdoors or Great Outdoors names, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Chapter 1. What's new in Planning Analytics Workspace

Find out what's new or changed in the most recent release of IBM Planning Analytics Workspace.The new features described here always reflect the full capabilities of Planning Analytics Workspace on cloud.

If you use Planning Analytics Workspace Local, some features might not be available. For more information, see Applicability of Planning Analytics Workspace documentation (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_prism_applicability_documentation.html).

2.0.57 - What's new, October 2, 2020

Planning Analytics Workspace 2.0.57 SC is a local-only release that includes significant changes in the look and feel of Planning Analytics Workspace, as well as important new functionality.

Updates to each version of Planning Analytics Workspace are cumulative. To see what was new in previous releases, see Chapter 1, "What's new in Planning Analytics Workspace," on page 1.

Learn from the experts

There are a lot of big changes in Planning Analytics Workspace 2.0.57. From a completely revised interface to significant new features like forecasting, the best place to learn about these changes is from the detailed blogs created by our subject matter experts.

Click here to visit the central blog about Planning Analytics Workspace enhancements and new capabilities. From this central blog, you'll find links to other blog posts and additional information from a range of Planning Analytics Workspace pros.

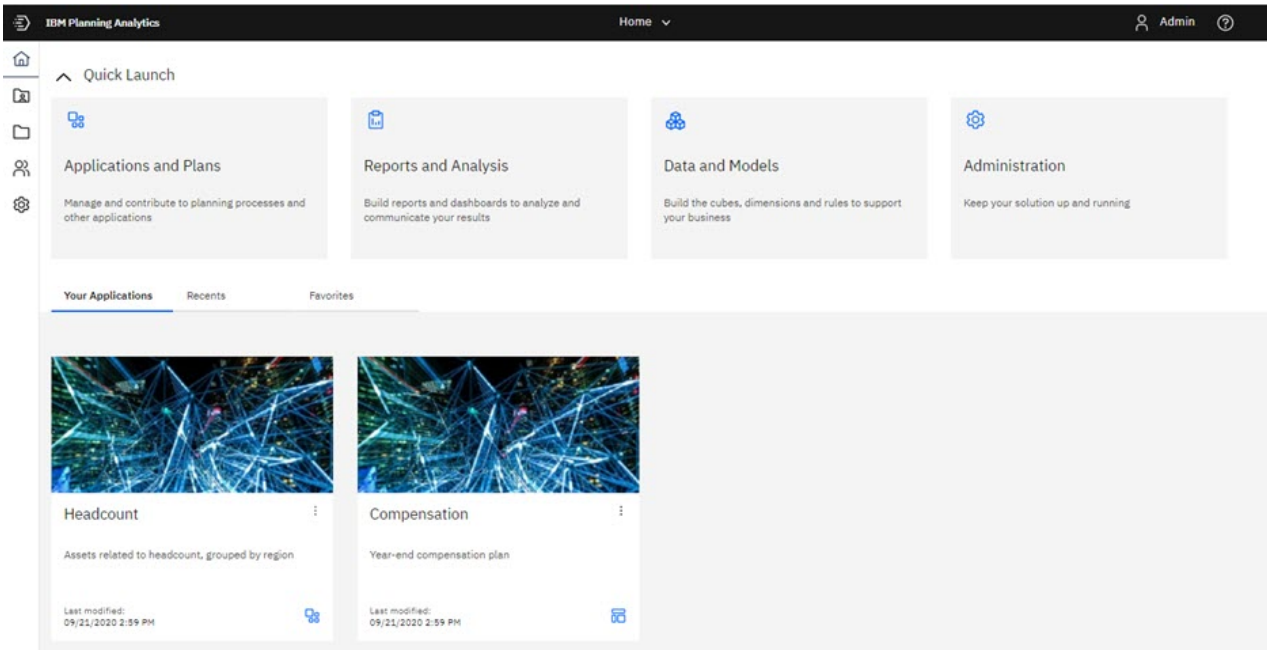
Improved look and feel

The Planning Analytics Workspace user interface has been updated to make it easier to accomplish tasks, provide a more consistent experience with other IBM products, and to simplify the transition between Planning Analytics Workspace and Cognos Analytics.

As part of the interface update, new icons adhering to the Carbon Design principles have been introduced throughout Planning Analytics Workspace. For more information about Carbon Design principles, see https://www.carbondesignsystem.com.

Note: Because the interface that exists in versions of Planning Analytics Workspace prior to 2.0.57 SC is still supported, it is necessary to differentiate between the new interface and the 'old' interface in the documentation. When there is a divergence of procedures or capabilities between the current (new) interface and the old, the documentation describes the current interface as Planning Analytics Workspace and describes the interface in 2.0.55 SC and prior versions as Planning Analytics Workspace Classic.

A new **Home** page for Planning Analytics Workspace is the first significant changes you'll notice. On the new **Home** page, you can quickly access the area you want to work in, customized for your role within Planning Analytics Workspace. You can also quickly open your applications and plans, as well as your recent and favorite items.



Manage and contribute to planning processes and other applications

Build reports and dashboards to analyze and communicate your results

Build the cubes, dimensions and rules to support your business

Keep your solution up and running

Assets related to headcount, grouped by region

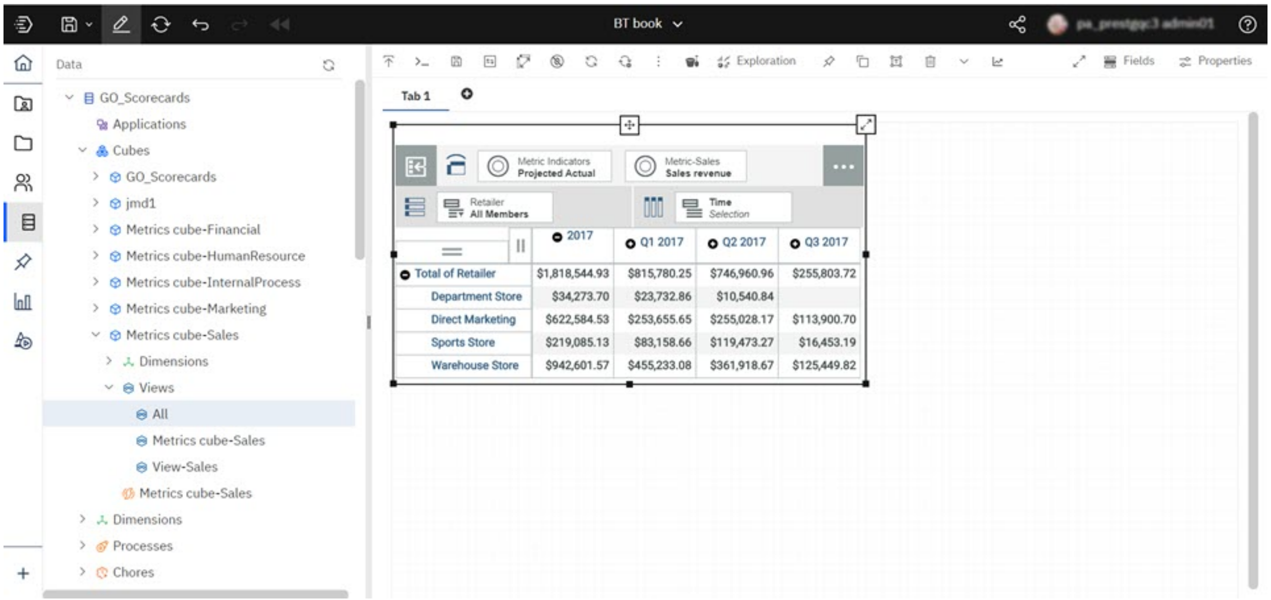
Last modified: 09/21/2020 2:59 PM

Year-end compensation plan

Last modified: 09/21/2020 2:59 PM

The prominent Quick Launch tiles that provide immediate access to **Applications and Plans**, **Reports and Analysis**, **Data and Models**, and **Administration** are dynamically displayed depending upon your role when you log in to Planning Analytics Workspace. Only the tiles that you can use based on your role appear. For example, an administrator sees all of the Quick Launch tiles, while an analyst sees only **Applications and Plans** and **Reports and Analysis**. You can click the Quick Launch show/hide button ^ Quick Launch to show or hide the Quick Launch tiles. When you hide the tiles, you can see more of your applications, recent items, or favorite items.

You'll also notice improvements in other familiar places. For example, in books the user interface is simplified, new icons are present, and the toolbar has been decoupled from the view.



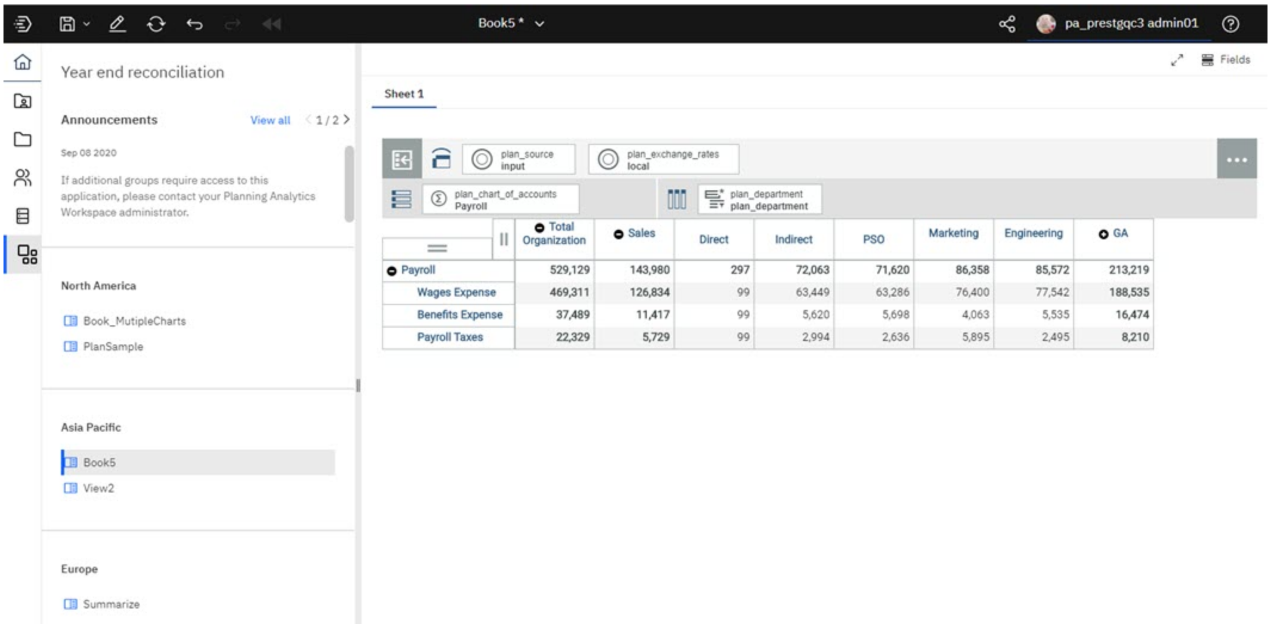
	2017	Q1 2017	Q2 2017	Q3 2017
Total of Retailer	$1,818,544.93	$815,780.25	$746,960.96	$255,803.72
Department Store	$34,273.70	$23,732.86	$10,540.84	
Direct Marketing	$622,584.53	$253,655.65	$255,028.17	$113,900.70
Sports Store	$219,085.13	$83,158.66	$119,473.27	$16,453.19
Warehouse Store	$942,601.57	$455,233.08	$361,918.67	$125,449.82

Use of the new interface is described in the relevant topics throughout the Planning Analytics Workspace help and documentation.

Applications and Plans

Applications and plans let you organize logically related Planning Analytics Workspace assets such as books, view, and websheets in containers.

An application contains related assets that are grouped in sections. These sections might reflect the structure of your organization, planning and budgeting requirements, or any other relevant grouping of assets. While an application contains logically related assets, there are no implied or required actions associated with the assets or sections in an application. An individual asset can belong to more than one application.

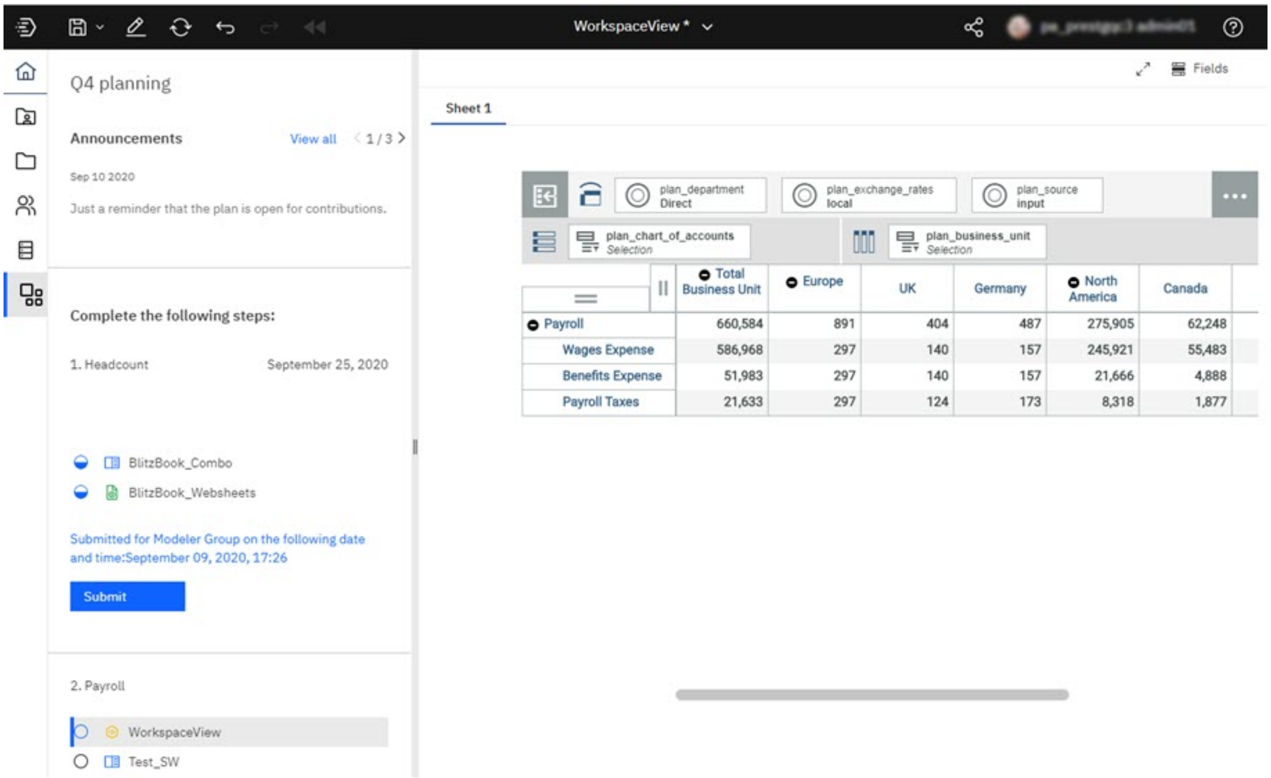


Sep 08 2020

If additional groups require access to this application, please contact your Planning Analytics Workspace administrator.

	Total Organization	Sales	Direct	Indirect	PSO	Marketing	Engineering	GA
Payroll	529,129	143,980	297	72,063	71,620	86,358	85,572	213,219
Wages Expense	469,311	126,834	99	63,449	63,286	76,400	77,542	188,535
Benefits Expense	37,489	11,417	99	5,620	5,698	4,063	5,535	16,474
Payroll Taxes	22,329	5,729	99	2,994	2,636	5,895	2,495	8,210

A plan contains assets that are grouped in steps. These steps can represent discrete tasks or contributions that must be completed in a planning or budgeting process. While steps can be ordered in a plan, there is no requirement for contributors to complete the steps sequentially; they can be completed in any order. Steps can also be assigned a due date for contributions. An administrator can require that steps be explicitly submitted for approval, and an administrator can reject and reset a submission.


WorkspaceView *

[redacted]

Q4 planning

Announcements View all 1 / 3

Sep 10 2020

Just a reminder that the plan is open for contributions.

Complete the following steps:

1. Headcount September 25, 2020

BlitzBook_Combo

BlitzBook_Websheets

Submitted for Modeler Group on the following date and time:September 09, 2020, 17:26

Submit

2. Payroll

WorkspaceView

Test_SW

plan_department Direct | plan_exchange_rates local | plan_source input

plan_chart_of_accounts *Selection* | plan_business_unit *Selection*

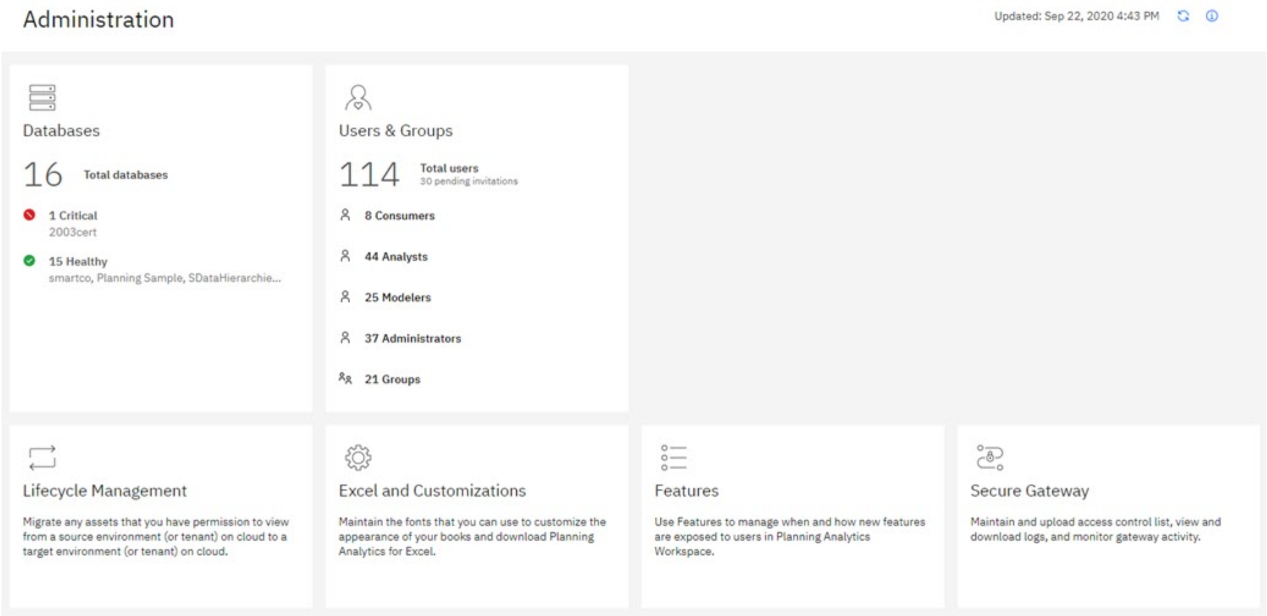
	Total Business Unit	Europe	UK	Germany	North America	Canada
Payroll	660,584	891	404	487	275,905	62,248
Wages Expense	586,968	297	140	157	245,921	55,483
Benefits Expense	51,983	297	140	157	21,666	4,888
Payroll Taxes	21,633	297	124	173	8,318	1,877

For complete details on creating, managing, and using applications and plans, see Applications and plans.

Administration page changes

The Planning Analytics Workspace **Administration** page has been reorganized to provide greater insight into your environment and to simplify access to administrative tasks.

The Administration page now includes several task-specific tiles. The Databases tile provide quick insight into the health of your databases. The Users and Groups tile lets you know how many users are assigned to each role and the number of groups that are defined.


Administration

Updated: Sep 22, 2020 4:43 PM

Databases

16 Total databases

1 Critical
2003cert

15 Healthy
smartco, Planning Sample, SDataHierarchie...

Users & Groups

114 Total users
30 pending invitations

8 Consumers

44 Analysts

25 Modelers

37 Administrators

21 Groups

Lifecycle Management

Migrate any assets that you have permission to view from a source environment (or tenant) on cloud to a target environment (or tenant) on cloud.

Excel and Customizations

Maintain the fonts that you can use to customize the appearance of your books and download Planning Analytics for Excel.

Features

Use Features to manage when and how new features are exposed to users in Planning Analytics Workspace.

Secure Gateway

Maintain and upload access control list, view and download logs, and monitor gateway activity.

You can click any tile to perform the administrative tasks associated with the tile.

The tiles available on the **Administration** page vary depending on whether you are running Planning Analytics Workspace Local or on Cloud. The Agents tile is available only on Local, while the Secure Gateway tile is available only on Cloud.

For complete details on using the new Administration page, see Administer Planning Analytics Workspace.

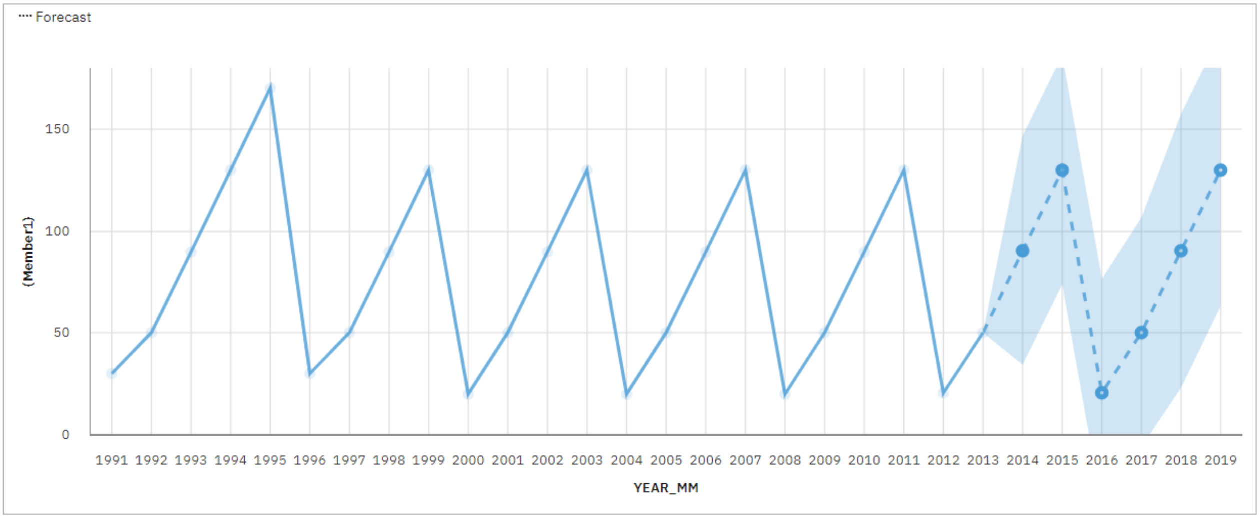
Forecasting

You can now use forecasting in IBM Planning Analytics Workspace to discover and model trend, seasonality, and time dependence in data.

You can forecast in Planning Analytics Workspace by using automated tools that model time-dependent data. Automated model selection and tuning makes forecasting easy to use, even if you are not familiar with time series modeling.

Forecasts and their confidence bounds are displayed in visualizations as a continuation of historic data. You can also view the statistical details for generated models if you want to see the technical background.

The following example shows forecasting values and confidence bounds in a line visualization.



Specifying time series in forecasts often requires data manipulation. Planning Analytics Workspace supports a wide range of time series without the need for manipulation, ranging from standard date and time types, to nested periodic and cyclical time fields. When data is recognized as a time series, data preparation is automated. Appropriate trend and seasonal periods are detected, and models are selected from a set of nine different model types.

You can forecast in line, bar, and column visualizations. Forecasting allows analysis of hundreds of time series per visualization. Forecasts and confidence bounds are computed for each time series, and displayed in the visualization as extensions of the current data. You can inspect each time series separately, and tailor the forecast and results to your own data and requirements.

If you are familiar with forecasting models, you can view the selected model type, estimated model parameters, standard accuracy measures, and processing summary information.

For complete details on using forecasting in Planning Analytics Workspace, including a complete tutorial, see Forecasting.

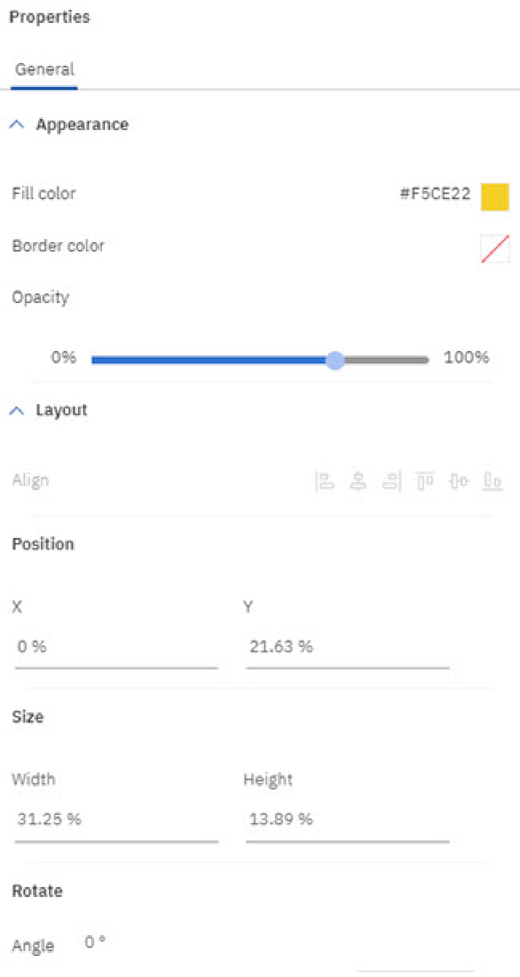
Book and visualization improvements

The properties available to manage all aspects of your books have been significantly expanded. The visualization types available to use in Planning Analytics Workspace have been updated to provide more and improved visualizations.

Books

All objects in a book, such as explorations, visualizations, buttons, images, and text, can now be precisely managed using an expanded selection of properties.

When you select an object in a book and then click the **Properties** tab, you'll see an expanded list of properties that you can set to manage the object in your book. The properties available vary by object type, and the properties you're familiar with are still available, but new properties allow you to precisely manage the size, position, alignment, and appearance of all objects in a book.



Properties

General

Appearance

Fill color #F5CE22

Border color

Opacity

0% 100%

Layout

Align

Position

X 0 %

Y 21.63 %

Size

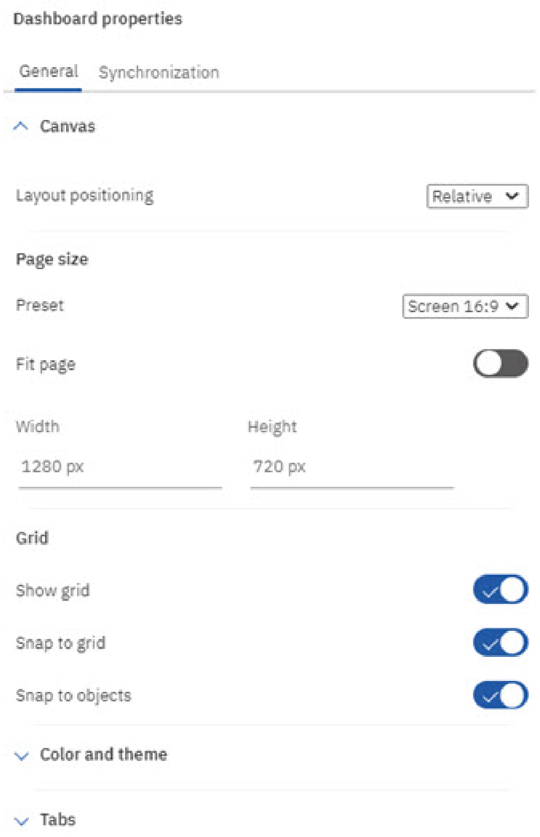
Width 31.25 %

Height 13.89 %

Rotate

Angle 0 °

You can also set Dashboard properties to manage the general appearance of your book.



Dashboard properties

General | Synchronization

Canvas

Layout positioning: Relative

Page size

Preset: Screen 16:9

Fit page: off

Width: 1280 px

Height: 720 px

Grid

Show grid: on

Snap to grid: on

Snap to objects: on

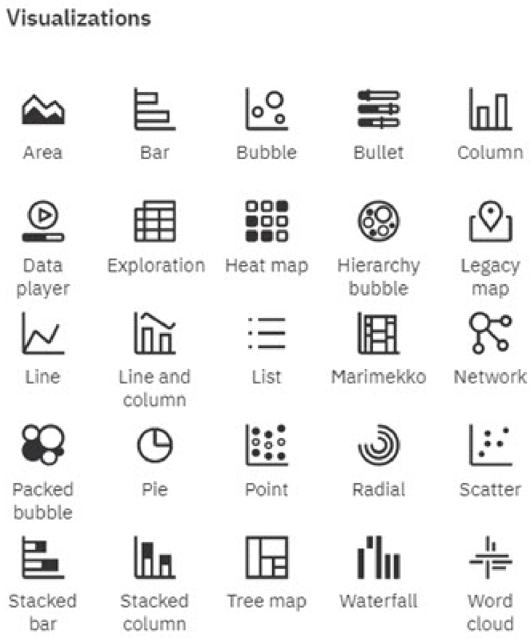
Color and theme

Tabs

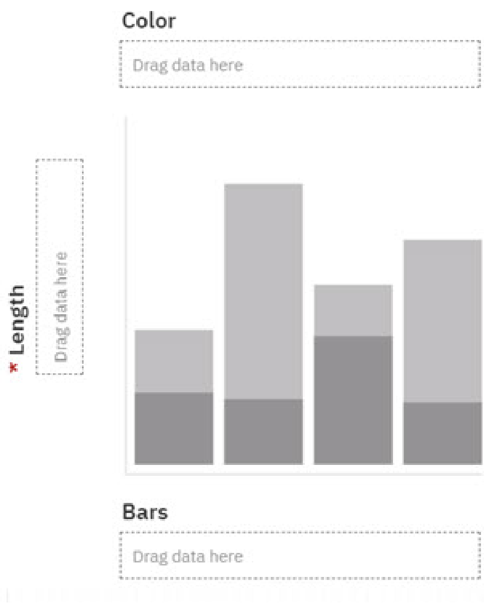
Details on using these properties are provided in the individual topics that describe how to build and manage books. For more information, see Work in books and views.

Visualizations

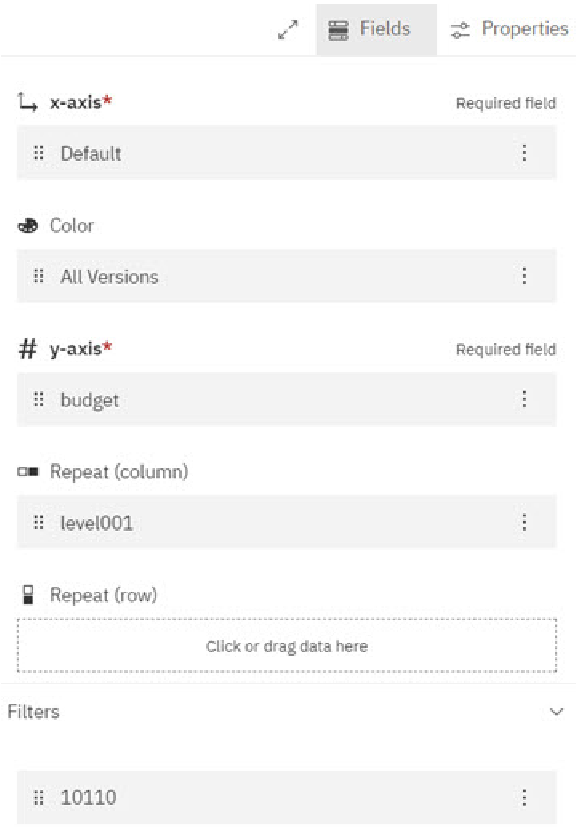
The list of visualizations available in Planning Analytics Workspace has been expanded to include twenty five options.



Explorations can be changed to visualizations in the same way you've always done it, but you can also place an empty visualization onto a book and use drop zones to build the visualization.



You can drag and drop dimensions from the Data tree onto the **Drag data here** drop zones to build a visualization from scratch. Once the visualization is complete, you can manage the visualization using the Fields tab on a book. You can drag and drop dimensions to different fields to change the structure of your visualization or you can click a field to select a new member to use in your visualization.



x-axis* Required field

Default

Color

All Versions

y-axis* Required field

budget

Repeat (column)

level001

Repeat (row)

Click or drag data here

Filters

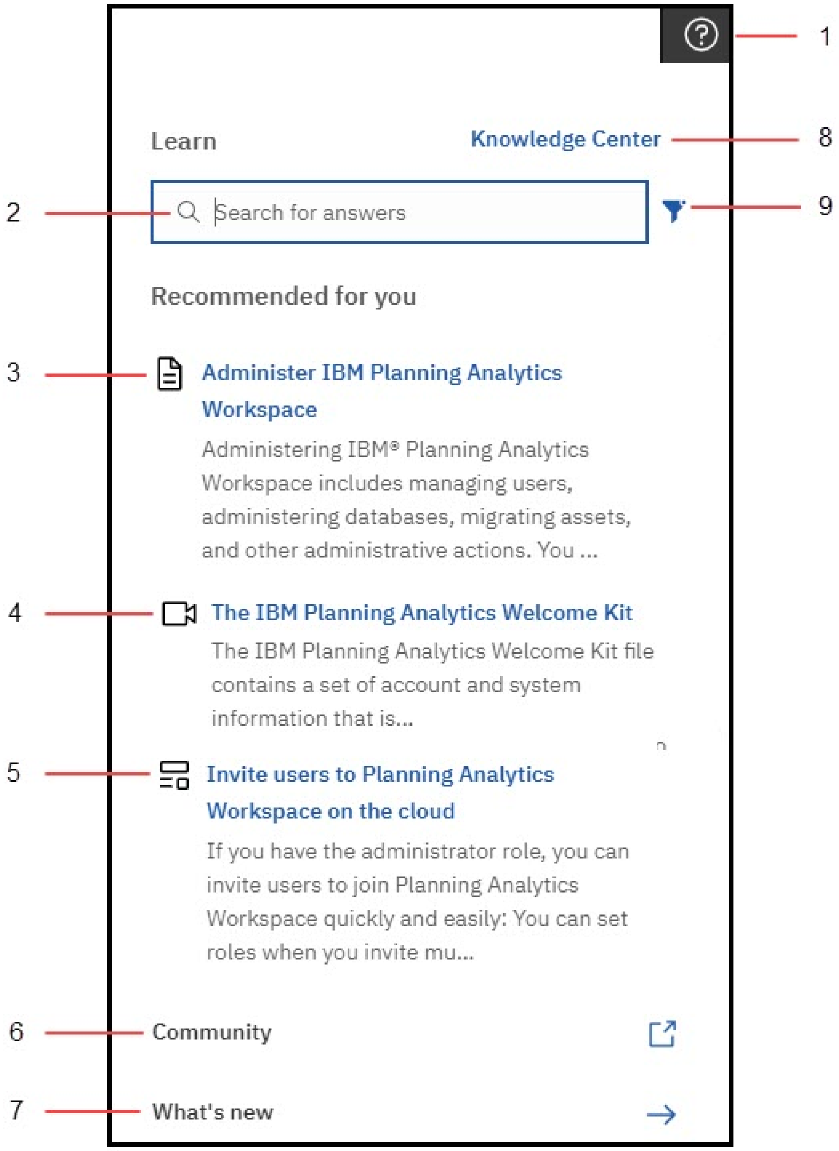
10110

For further details, see Visualizations.

Ask for help in the cognitive Learn pane

If you want to learn more about how to use Planning Analytics Workspace, click the Help icon ⓘ and ask a question. The cognitive help tailors your help experience based on where you are working in the product and finds only the answers that are relevant to your user role. You can find the latest videos, blogs, and documentation.

Use these features on the Learn pane to find answers and be more productive!



Learn

Knowledge Center

Search for answers

Recommended for you

Administer IBM Planning Analytics Workspace

Administering IBM® Planning Analytics Workspace includes managing users, administering databases, migrating assets, and other administrative actions. You ...

The IBM Planning Analytics Welcome Kit

The IBM Planning Analytics Welcome Kit file contains a set of account and system information that is...

Invite users to Planning Analytics Workspace on the cloud

If you have the administrator role, you can invite users to join Planning Analytics Workspace quickly and easily: You can set roles when you invite mu...

Community

What's new

1

Click the Help icon to open the Learn pane (it remembers where you were the last time you opened it). Click anywhere to close it. The Learn pane recommends content that relates to your task and finds similar content that you might also like. And, it is always learning! When you search and find answers, you are training the Learn pane and those answers influence future recommendations.

2

Type a question in the Search box. You can search in any supported language in the Learn pane and you see translated documentation in your search results. You also see blogs and videos that match your search, however, blogs and videos aren't translated.

3

Read the formal product documentation, sourced from IBM Knowledge Center.

4

Watch a video! Sometimes the best way to learn is to see it in action.

5

Read a post in the Planning Analytics Community Blog The community blog posts are written by experts who use Planning Analytics Workspace and share their tips and tricks.

6

Go to the Planning Analytics Community. In the community, you can find the latest articles, blog posts, and events. You can also start and contribute to discussions about Planning Analytics.

7

Click **What's New** to find out what is new in the latest release of Planning Analytics Workspace.

8

Visit the IBM Knowledge Center for all IBM Planning Analytics documentation In the IBM Knowledge Center, you can read all documentation, including related products.

9

Filter your search results to show only your preferred content type: videos, blogs, or documentation.

Considerations for upgrading to Planning Analytics Workspace 2.0.57 SC

Users should be aware of the following considerations before upgrading to the new experience Planning Analytics Workspace 2.0.57 SC.

The upgrade to Planning Analytics Workspace 2.0.57 SC is a permanent upgrade. There is no way to revert to a prior version of Planning Analytics Workspace Classic.

Features not supported in initial 2.0.57 SC release

These features are not supported in the initial new experience Planning Analytics Workspace 2.0.57 SC release. Support for these features may be reintroduced in subsequent releases.

- Metrics charts/Scorecarding (Impact Diagram, Strategy Map, Custom Diagram) are not supported.
- Tree map visualizations upgrade successfully *except* when more than one dimension is present on an axis, in which case the tree map is converted to an Exploration during book upgrade.
- The Reports and Analysis landing page does not display individual tiles for books, views, or websheets. Rather, the Reports and Analysis landing page displays a searchable and sortable list of all assets.
- Mobile devices are not fully supported in the initial new experience release, as some gestures are not yet implemented.
- An Administrator cannot set up a global color palette.
- When you click **Share** > **Export**, you cannot share a book or view as an image or PowerPoint document. You can, however, export as PDF with enhanced print options.
- The Intent bar (sometimes called the NLP bar) is not available in this release.

Differences in behavior between Planning Analytics Workspace 2.0.57 SC and Planning Analytics Workspace Classic

- Chats are deprecated, as previously announced in this deprecation notice.
- Bookmarks and history are no longer available on the Content tree/Data tree. Instead, you can use the Recents or Favorites tabs on the Planning Analytics Workspace Home page to open assets that you have recently viewed or favorited.
- Collections has been changed to Pins. You can pin a view or websheet from a book. You can access pinned items from the Pin 📌 button while in Edit mode.

2.0.55 - What's new, August 11, 2020

This version of Planning Analytics Workspace on cloud includes fixes only.

Updates to each version of Planning Analytics Workspace are cumulative. To see what was new in the previous release, see “2.0.54 - What's new, July 15, 2020” on page 11.

2.0.54 - What's new, July 15, 2020

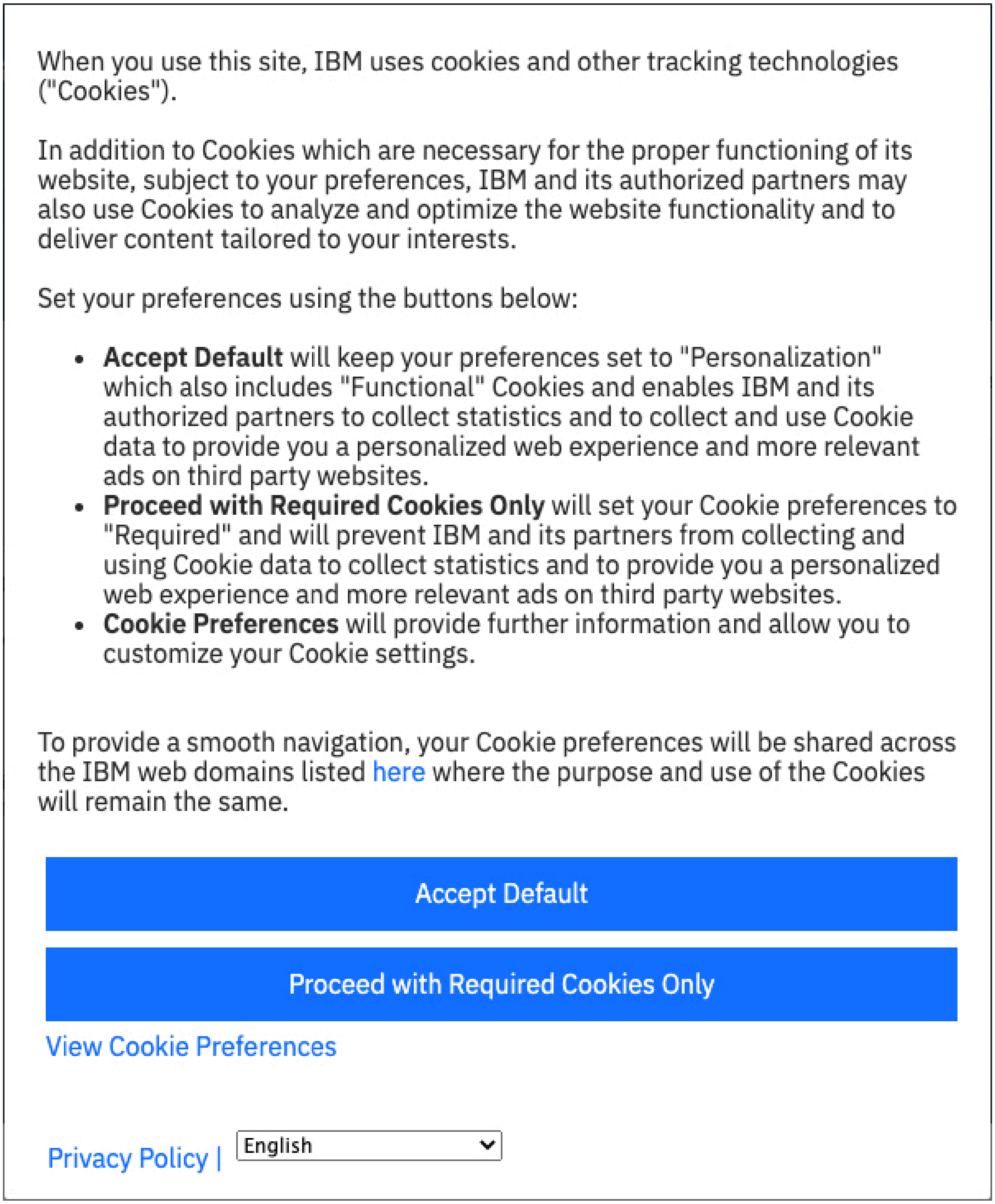
See what's new in version 2.0.54 of IBM Planning Analytics Workspace in the following topics.

Note: Planning Analytics Workspace on cloud is available from July 15, 2020. Planning Analytics Workspace local is available starting from July 12, 2020.

Set your cookie preferences

IBM begins to capture metrics of your usage of IBM Planning Analytics Workspace on cloud to improve the product, and its capabilities, with the release of 2.0.54.

If you are based in the European Union, you will see a prompt the first time that you log on after Planning Analytics Workspace on cloud is upgraded. Users are determined to be in the European Union based on their IP address by using geolocation services. Your acceptance of cookie preferences is stored as a cookie in your browser, so if you clear your cookies or change browser, you see the prompt again. This is standard behavior, and enables you to control the level of cookies that you allow to be stored.



When you use this site, IBM uses cookies and other tracking technologies ("Cookies").

In addition to Cookies which are necessary for the proper functioning of its website, subject to your preferences, IBM and its authorized partners may also use Cookies to analyze and optimize the website functionality and to deliver content tailored to your interests.

Set your preferences using the buttons below:

- **Accept Default** will keep your preferences set to "Personalization" which also includes "Functional" Cookies and enables IBM and its authorized partners to collect statistics and to collect and use Cookie data to provide you a personalized web experience and more relevant ads on third party websites.
- **Proceed with Required Cookies Only** will set your Cookie preferences to "Required" and will prevent IBM and its partners from collecting and using Cookie data to collect statistics and to provide you a personalized web experience and more relevant ads on third party websites.
- **Cookie Preferences** will provide further information and allow you to customize your Cookie settings.

To provide a smooth navigation, your Cookie preferences will be shared across the IBM web domains listed here where the purpose and use of the Cookies will remain the same.

If you are not based in the European Union, you are not prompted for cookies but you can select the level of cookies by clicking 👤 and then selecting **Cookie Preferences**. You can see more information about the levels of cookies by clicking **View Cookie Preferences** in the **Cookies** window.

To learn more, see:

Cookie preferences for Planning Analytics on cloud (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_paw_cookies.html)

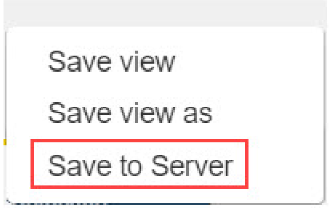
Save a view to the TM1 database

You can save a view from Planning Analytics Workspace to the TM1 database.

When a view is saved to the database, it can be used by TM1 processes as a data source from which you can extract data and create or update objects or data.

A view saved to the TM1 database is also available to any Planning Analytics client that connects to the database.

To save a view to the TM1 database, select **Save to Server** from the shortcut bar.



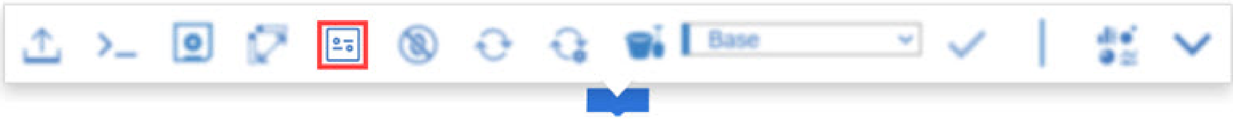
To learn more, see:

Save a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_save_view.html).

Format numbers for an entire view

You can apply number formatting for an entire view from the shortcut bar.

To apply formatting, click the **Format** icon on the shortcut bar and select the desired format.



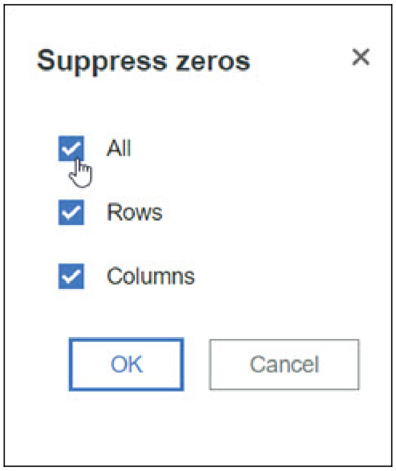
To learn more, see:

Change the format of data in a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_format_data.html).

Suppress zeros for an entire view

You can suppress zeros for an entire view from the shortcut bar.

To suppress all rows and columns that contain only zeros, click ⊘ on the shortcut bar, then select **All**.


Suppress zeros

☑ All

☑ Rows

☑ Columns

To learn more, see:

Suppress zeros (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_prism_suppress.html).

2.0.53 - What's new, May 21, 2020

Planning Analytics Workspace on cloud was refreshed on May 21, 2020 to include fixes only.

IBM Planning Analytics Workspace Distributed for Planning Analytics Workspace Local now supports Red Hat OpenShift deployment. For details, see Install on OpenShift.

Updates to each version of Planning Analytics Workspace are cumulative. For more information on what was new in the previous release, see “2.0.52 - What's new, April 22, 2020” on page 14.

2.0.52 - What's new, April 22, 2020

See what's new in version 2.0.52 of IBM Planning Analytics Workspace in the following topics.

Usability improvements in the set editor

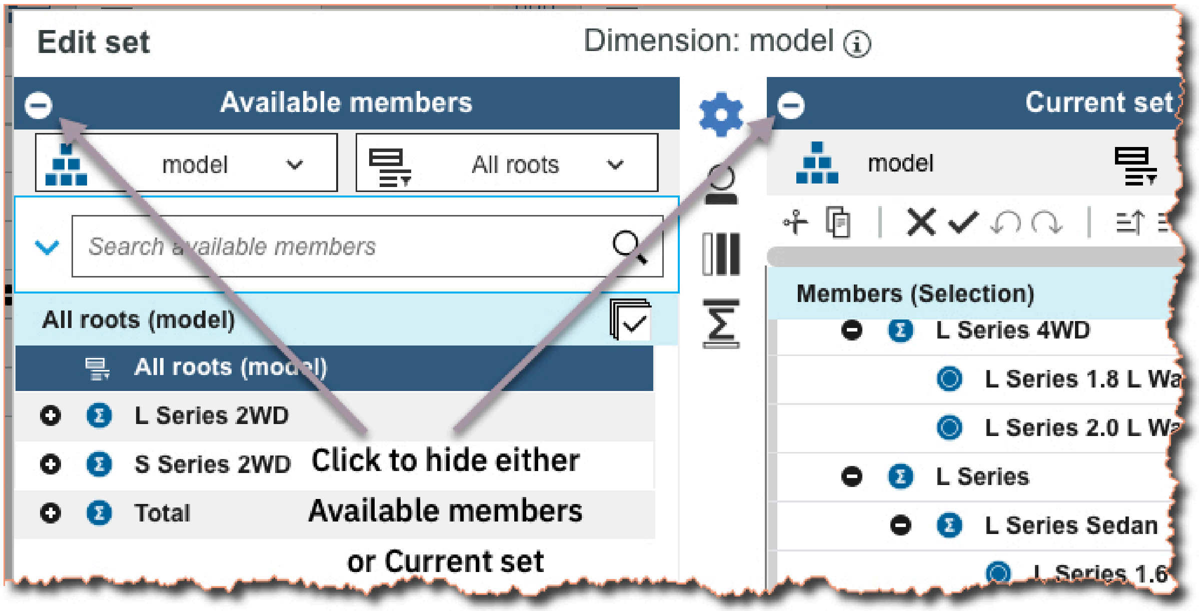
Quickly and easily define, edit, and re-order sets with new improvements to work with large dimensions.

This video demonstrates the new features in the set editor.

https://youtu.be/-69uMHYE7mA

Focus on one area at a time

You can hide the pane that you are not currently working in to enable you to focus on one area at a time.



Click to hide either Available members or Current set

Maximize the set editor

Quickly maximize the set editor by clicking the **Max** icon.

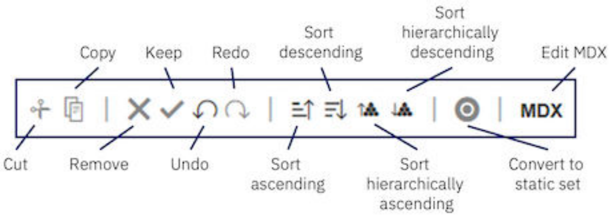
If you added the set editor from the tree, you can resize the set editor by dragging the grab handles.

Drag and drop

Drag members into the **Current set** from **Available members**. Dragging uses the default insert settings ◎. If there are a lot of members in the **Current set**, you can scroll down the pane while dragging the selection into the correct position.

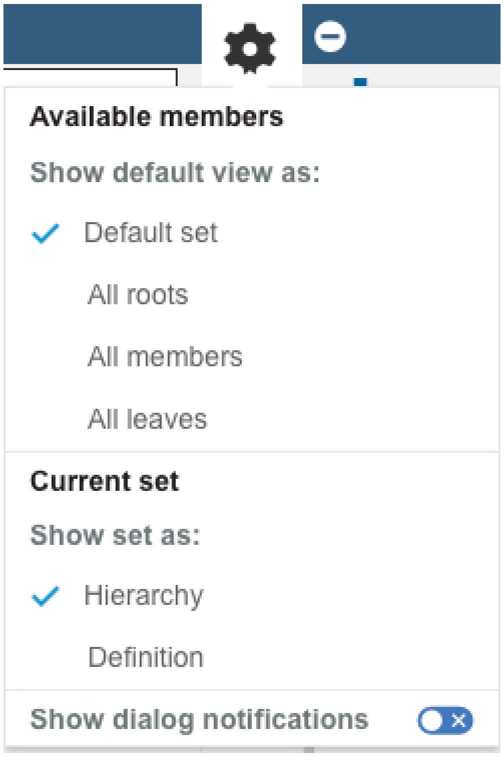
Easy access to functions

The new toolbar makes it easy to edit the members in the current set, with the most frequently used functions available.



Configure default view settings for the Available members and Current set panes

For the **Available members** pane, you can choose to display **Default set**, **All roots**, **All members**, or **All leaves** by default. For the **Current set** pane, you can choose whether to display the members as a **Hierarchy**, or as a **Definition** (MDX).



Available members

Show default view as:

✓ Default set

All roots

All members

All leaves

Current set

Show set as:

✓ Hierarchy

Definition

Show dialog notifications ○✕

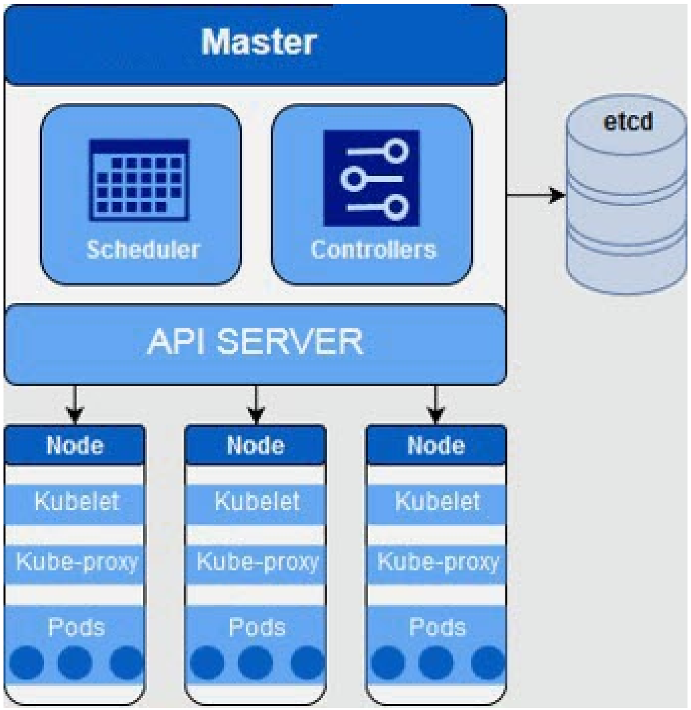
Click ⚙ to access the **Settings** menu.

To learn more, see:

Create and edit sets (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_create_sets.html)

Deploy Planning Analytics Workspace Distributed on Kubernetes for high availability (local only)

⌂ IBM Planning Analytics Workspace Distributed is an upgrade for Planning Analytics Workspace Local that can be deployed in a container orchestration engine for high availability, fail-over, scalability, and fault tolerance.



Note: Planning Analytics Workspace Distributed on Kubernetes is currently supported on Red Hat Enterprise Linux® (RHEL) only.

What's the difference between Planning Analytics Workspace Distributed and Planning Analytics Workspace Local single machine installation?

- Planning Analytics Workspace Distributed has the same set of microservices and capabilities as Planning Analytics Workspace Local.
- Planning Analytics Workspace Distributed has multiple instances of stateless services for availability.
- Databases are configured for fault tolerance in Planning Analytics Workspace Distributed.
- Planning Analytics Workspace Distributed has no **Status** tab in the Planning Analytics Workspace administration tool.

Migration to Planning Analytics Workspace Distributed by using Planning Analytics Workspace Local backup is a seamless upgrade.

1. Back up Planning Analytics Workspace Local.
2. Then, follow the instructions to restore on Planning Analytics Workspace Distributed.

To learn more, see Installing Planning Analytics Workspace Distributed (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_distributed_install_overview.html).

Download a log file that records changes to the availability status of your database

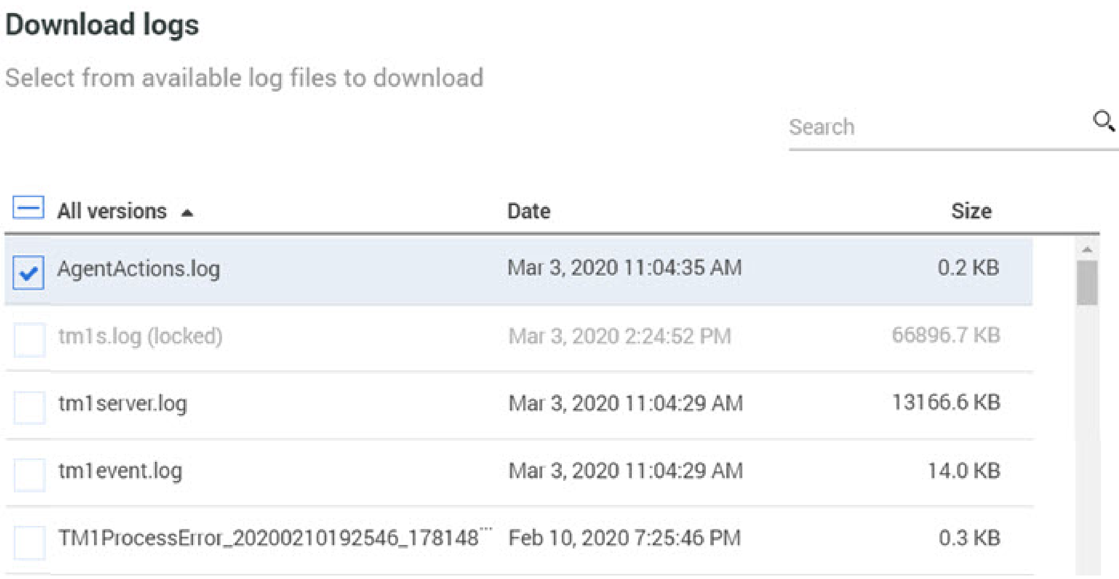
Use the AgentActions.log file to review the user and time associated with these database actions: start, stop, restart, and end process.

Every time a user performs an action that affects the availability of your Planning Analytics database, a record is written to the AgentActions.log file. This example shows instances of all the action types that are recorded in the log file.

```
2020-01-15 19:31 : User [admin01] performed [start] on the server.
2020-02-03 09:57 : User [admin01] performed [restart] on the server.
2020-03-05 20:10 : User [nolan] performed [stop] on the server.
2020-03-05 20:12 : User [nolan] performed [start] on the server.
2020-03-12 19:31 : User [admin01] performed [end process] on the server.
```

Note: The AgentActions.log file records only actions that are performed through the Planning Analytics Administration user interface. Crashes or actions performed through any means other than the user interface are not recorded.

To download the AgentActions.log file log, click ⋯ on a database tile and click **Download logs**. Then, select the AgentActions.log file and click **Download**.



Download logs

Select from available log files to download

Search

⊟ All versions ▲	Date	Size
☑ AgentActions.log	Mar 3, 2020 11:04:35 AM	0.2 KB
□ tm1s.log (locked)	Mar 3, 2020 2:24:52 PM	66896.7 KB
□ tm1server.log	Mar 3, 2020 11:04:29 AM	13166.6 KB
□ tm1event.log	Mar 3, 2020 11:04:29 AM	14.0 KB
□ TM1ProcessError_20200210192546_178148…	Feb 10, 2020 7:25:46 PM	0.3 KB

Note: ⌂ To enable the AgentActions.log on Planning Analytics Local, you must add the following parameter to the `bootstrap.properties` file for your Planning Analytics Administration agent:

`LOG_ACTIONS=TRUE`

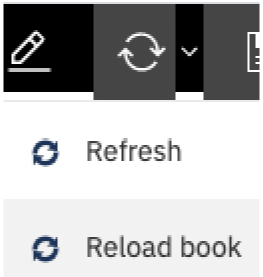
Restart the Planning Analytics Administration agent after you add the parameter.

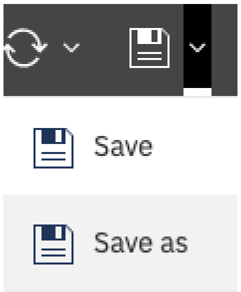
To learn more about the Planning Analytics Administration agent and the `bootstrap.properties` file, see Install and configure the Planning Analytics Administration agent.

To learn more about downloading log files, see Download database log files (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/paw_download_database_logs.html).

Quick access to Save as and Reload book actions

Now you can access **Save as** from the **Save** menu, and **Reload book** from the **Refresh** menu.





Previously, you accessed **Save as** and **Reload book** from ☰.

To learn more, see:

Refresh data (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_paw_refreshing_data.html)

New book icons

We have made some changes to the edit mode and personal menu icons. You will also see new home page and maximize screen icons.

✎	Turn edit mode on or off.
⌂	Go to the home page.
⤢	Maximize the screen.
👤	Access the personal menu.

2.0.50, 2.0.51

Planning Analytics Workspace version 2.0.50, 2.0.51, and 2.0.52, have been combined so that the release of Planning Analytics Workspace 2.0.52 aligns with the IBM Planning Analytics for Microsoft Excel release.

2.0.49 - What's new, February 19, 2020

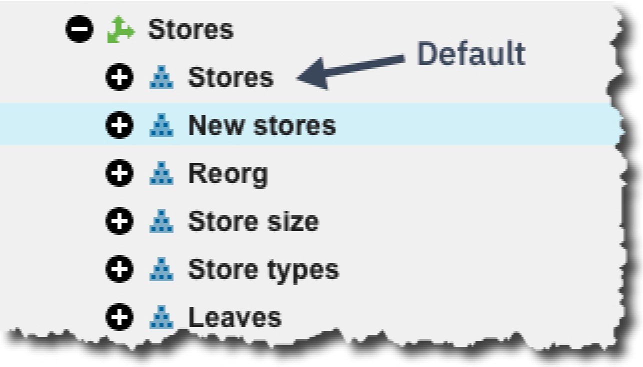
See what's new in version 2.0.49 of IBM Planning Analytics Workspace in the following topics.

Hierarchies are now sorted alphabetically

Hierarchies are now sorted alphabetically in the content tree, and in the set editor, so that the order of hierarchies is predictable.

The sort order for hierarchies is determined on the following basis:

- The default hierarchy (typically, the hierarchy with the same name as the dimension) appears first.
- The all-leaves hierarchy (typically called Leaves) appears last.
- All other hierarchies are sorted alphabetically by their caption.



To learn more, see:

Hierarchies (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_paw_modeling_hierarchies.html)

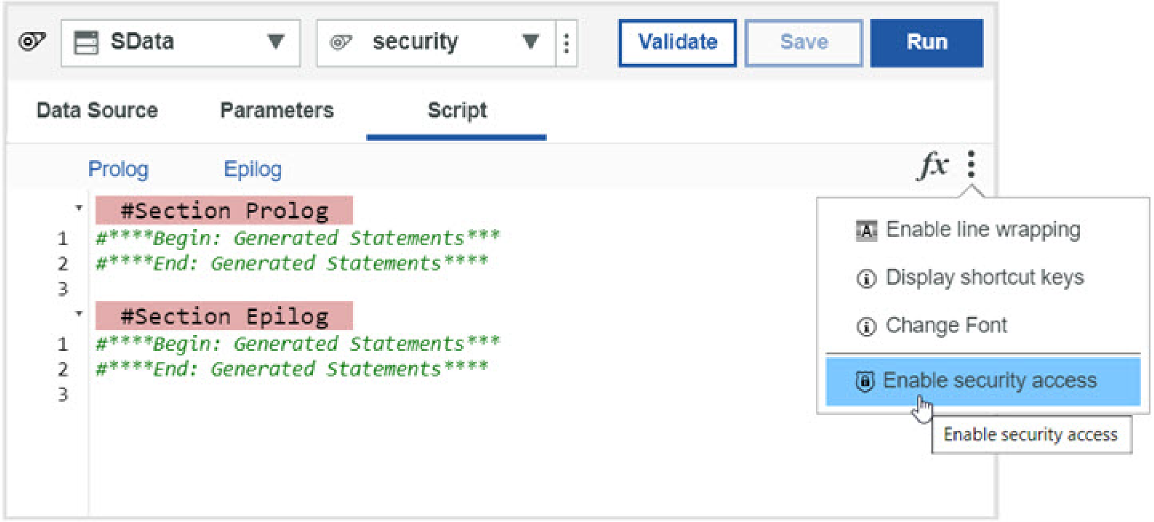
Enable security access for processes

By default, security access is disabled when you create a new process. This means the process cannot modify security data. If you want to allow a process to modify security, you must enable security access for that process.

Note that you must have an Administrator or Modeler role in Planning Analytics Workspace **and** be a member of either the ADMIN or SecurityAdmin group in TM1 to enable security access.

To enable security access:

On the **Script** tab of a process, click the **Options** button, then click **Enable security access**.



```
#Section Prolog
#****Begin: Generated Statements***
#****End: Generated Statements****

#Section Epilog
#****Begin: Generated Statements***
#****End: Generated Statements****

```

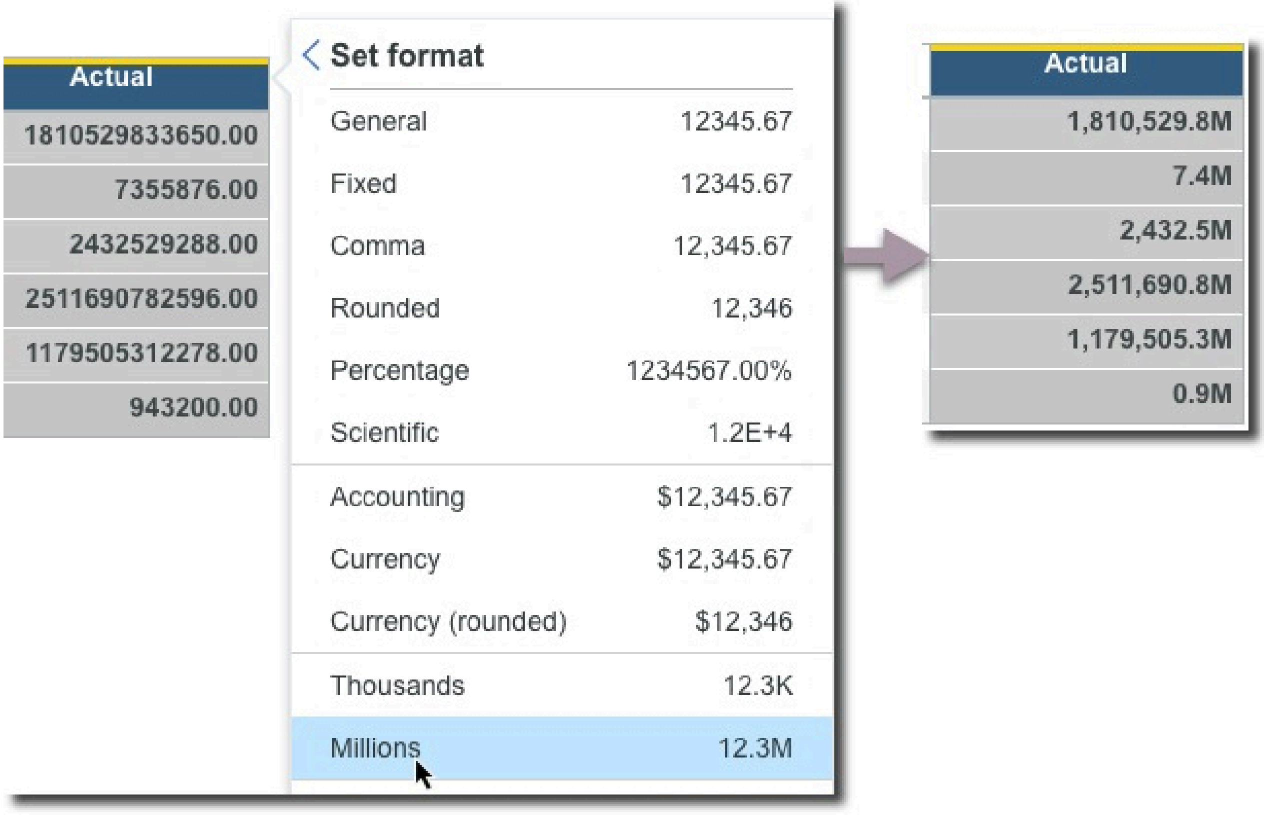
To learn more, see Processes (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_paw_processes.html).

2.0.48 - What's new, January 14, 2020

See what's new in version 2.0.48 of IBM Planning Analytics Workspace in the following topics.

Abbreviate numbers in views to thousands or millions

You can abbreviate numbers quickly in a view to thousands or millions to make your data easier to read.



Actual
1810529833650.00
7355876.00
2432529288.00
2511690782596.00
1179505312278.00
943200.00

Set format

General	12345.67
Fixed	12345.67
Comma	12,345.67
Rounded	12,346
Percentage	1234567.00%
Scientific	1.2E+4
Accounting	$12,345.67
Currency	$12,345.67
Currency (rounded)	$12,346
Thousands	12.3K
Millions	12.3M

Actual
1,810,529.8M
7.4M
2,432.5M
2,511,690.8M
1,179,505.3M
0.9M

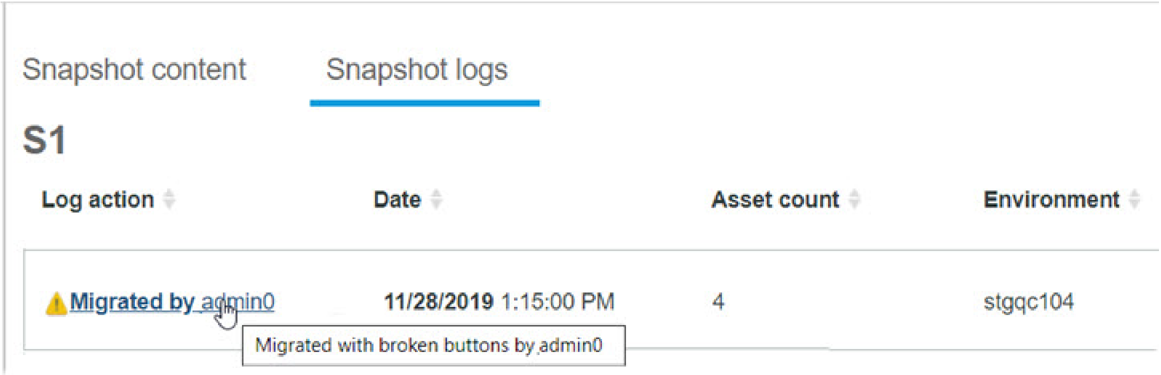
To learn more, see:

Change the format of data in a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_format_data.html)

Identify broken button links when you migrate a snapshot

When you migrate a book with a button, but do not migrate the target of the button, the button will be broken after migration. You can now view a list of migrated assets that result in broken buttons and take action to restore the buttons in the migration target environment.

When you migrate a snapshot that results in broken buttons, the **Snapshot logs** tab on the **Manage snapshots** page alerts you to this condition. A warning icon next to the **Migrated by** log action indicates the presence of broken buttons in the migrated snapshot. If you hover over the log action, an informational message confirms the issue.

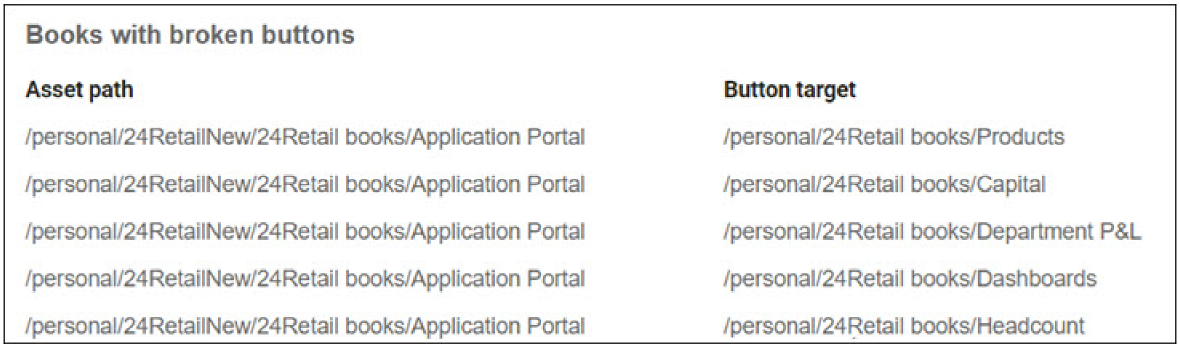


S1

Log action	Date	Asset count	Environment
⚠ Migrated by admin0	11/28/2019 1:15:00 PM	4	stgqc104

Migrated with broken buttons by admin0

When there are broken buttons in a migrated snapshot, the **Migrated by** log action becomes a link. Click the link to view a log report showing assets that contain broken buttons and the missing button targets.



Books with broken buttons

Asset path	Button target
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Products
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Capital
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Department P&L
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Dashboards
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Headcount

You can restore the broken buttons by adding the button targets to your snapshot and repeating the snapshot migration.

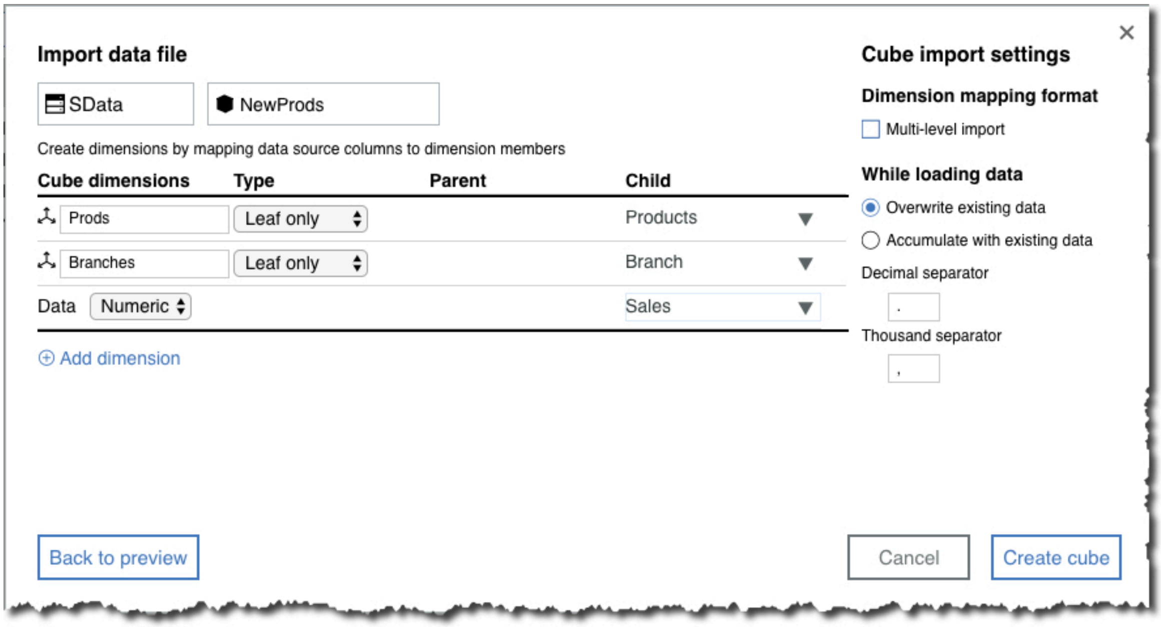
To learn more, see View snapshot logs (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/paw_asset_view_activity.html).

2.0.47 - What's new, November 20, 2019

See what's new in version 2.0.47 of IBM Planning Analytics Workspace in the following topics.

Create a cube by importing a text file

You can quickly and easily create and populate a cube and its dimensions without having to write processes, by importing a text file into IBM Planning Analytics Workspace.



Import data file

SData | NewProds

Create dimensions by mapping data source columns to dimension members

Cube dimensions	Type	Parent	Child
Prods	Leaf only		Products
Branches	Leaf only		Branch
Data	Numeric		Sales

Cube import settings

Dimension mapping format

□ Multi-level import

While loading data

● Overwrite existing data

○ Accumulate with existing data

Decimal separator: .

Thousand separator: ,

To learn more, see Create a cube from a file (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_cube_from_file.html).

Improvements to importing into dimensions

Improvements to importing into dimensions means that you can easily update an existing dimension hierarchy, and have more control over the dimension structure.

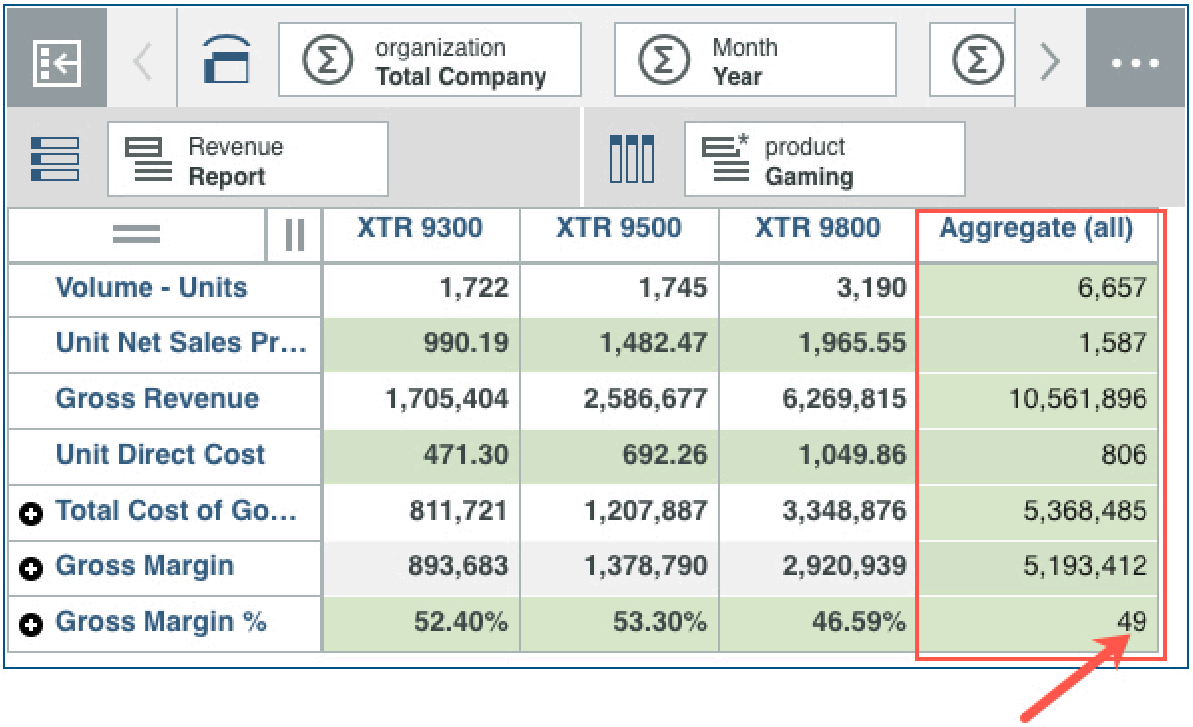
You can now update an existing dimension hierarchy (previously you could import only into an empty hierarchy). You can select a multi-level dimension structure when you create a new dimension by importing a text file, so you can quickly create a dimension with multiple levels. You can also specify a decimal separator and a thousand separator.

To learn more, see Import members and attributes into a dimension (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_drag_and_drop_files_to_create_members.html).

Apply aggregate calculations in a view

You can apply an aggregate calculation to two or more values in a view so that you can represent the aggregate value in a chart or report.

You can use aggregate calculations to summarize values that can't simply be added up or averaged. For example, percentage or ratio values that are the result of a rule calculation.



organization
Total Company

Month
Year

Revenue
Report

product
Gaming

	XTR 9300	XTR 9500	XTR 9800	Aggregate (all)
Volume - Units	1,722	1,745	3,190	6,657
Unit Net Sales Pr...	990.19	1,482.47	1,965.55	1,587
Gross Revenue	1,705,404	2,586,677	6,269,815	10,561,896
Unit Direct Cost	471.30	692.26	1,049.86	806
⊕ Total Cost of Go...	811,721	1,207,887	3,348,876	5,368,485
⊕ Gross Margin	893,683	1,378,790	2,920,939	5,193,412
⊕ Gross Margin %	52.40%	53.30%	46.59%	49

For both Planning Analytics Workspace on cloud and Planning Analytics Workspace Local, you must have IBM Planning Analytics version 2.0.9 or later.

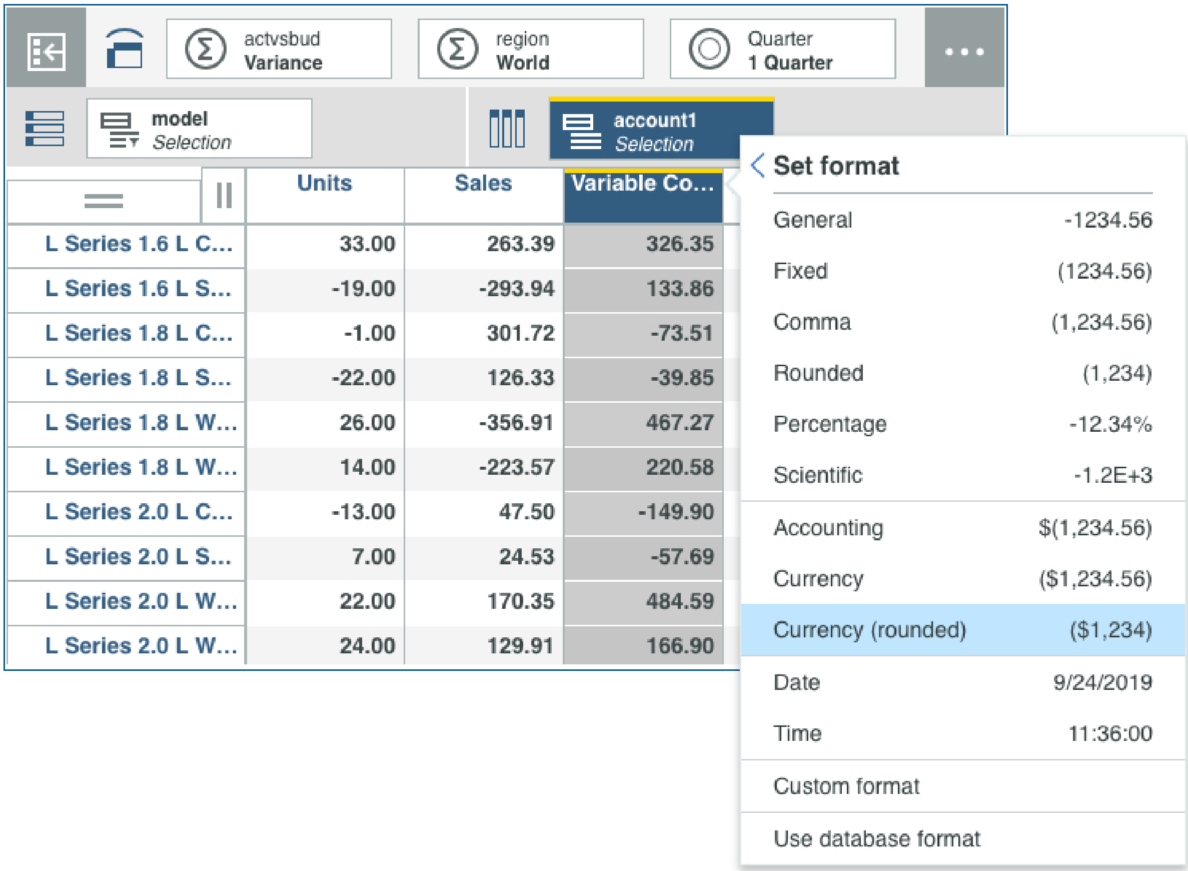
To learn more, see Adding a member calculation to a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_adding_a_member_calculation.html) and Adding a summary calculation to a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_adding_a_dynamic_calculation.html.

2.0.46 - What's new, October 15, 2019

See what's new in version 2.0.46 of IBM Planning Analytics Workspace in the following topics.

Quickly set the data format in the view

You can now set the format of data by column and row directly in the cell view, making it quicker and easier to change the formatting in a view.



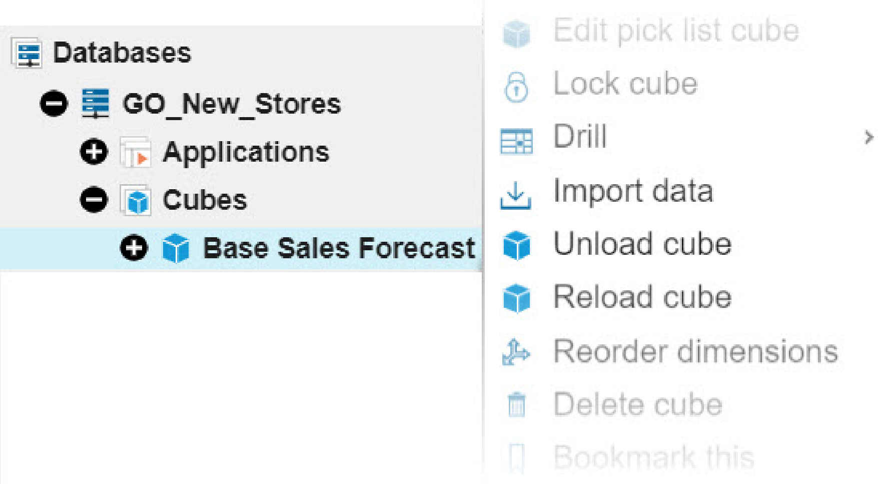
	Units	Sales	Variable Co...
L Series 1.6 L C...	33.00	263.39	326.35
L Series 1.6 L S...	-19.00	-293.94	133.86
L Series 1.8 L C...	-1.00	301.72	-73.51
L Series 1.8 L S...	-22.00	126.33	-39.85
L Series 1.8 L W...	26.00	-356.91	467.27
L Series 1.8 L W...	14.00	-223.57	220.58
L Series 2.0 L C...	-13.00	47.50	-149.90
L Series 2.0 L S...	7.00	24.53	-57.69
L Series 2.0 L W...	22.00	170.35	484.59
L Series 2.0 L W...	24.00	129.91	166.90

You can select a built in format, such as fixed, or you can create a custom format, if, say you wanted to use a specific currency symbol.

To learn more, see Change the format of data in a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_format_data.html) and Add cell values to the sheet as a cell view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_cell_widget.html).

Unload a cube from memory

You can unload a cube from memory to temporarily reduce RAM consumption or to assist in the development and troubleshooting of rules feeders. Unloading a cube also unloads any views of the cube from memory.



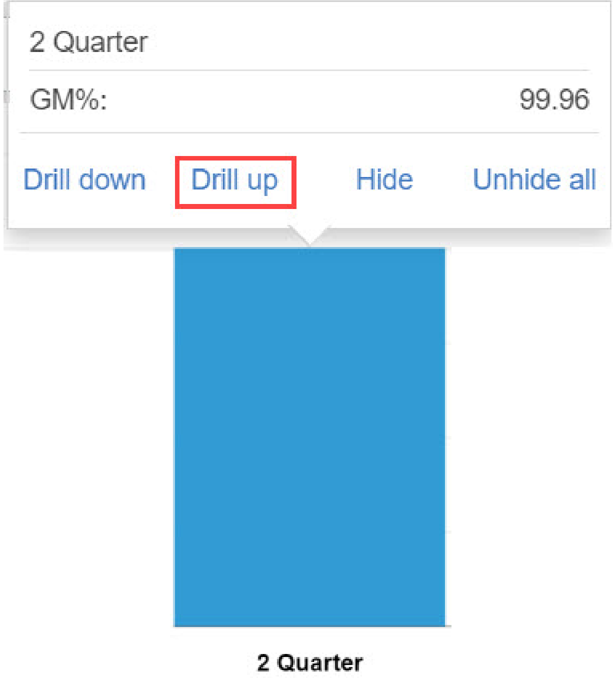
While a cube is unloaded, any request for data in the cube will cause the cube to be automatically reloaded, maintaining data availability. Additionally, you can use the **Reload** option to manually reload a cube.

You must be an administrator or modeler to unload or reload cubes.

To learn more, see Unload and reload a cube from memory (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_unload_cube.html).

Drill up on visualizations

You can now drill up to reveal the parent of a member in a visualization. In previous versions of Planning Analytics Workspace, you had to use the **Undo** option to view the parent of a member to which the **Drill down** option had been applied.



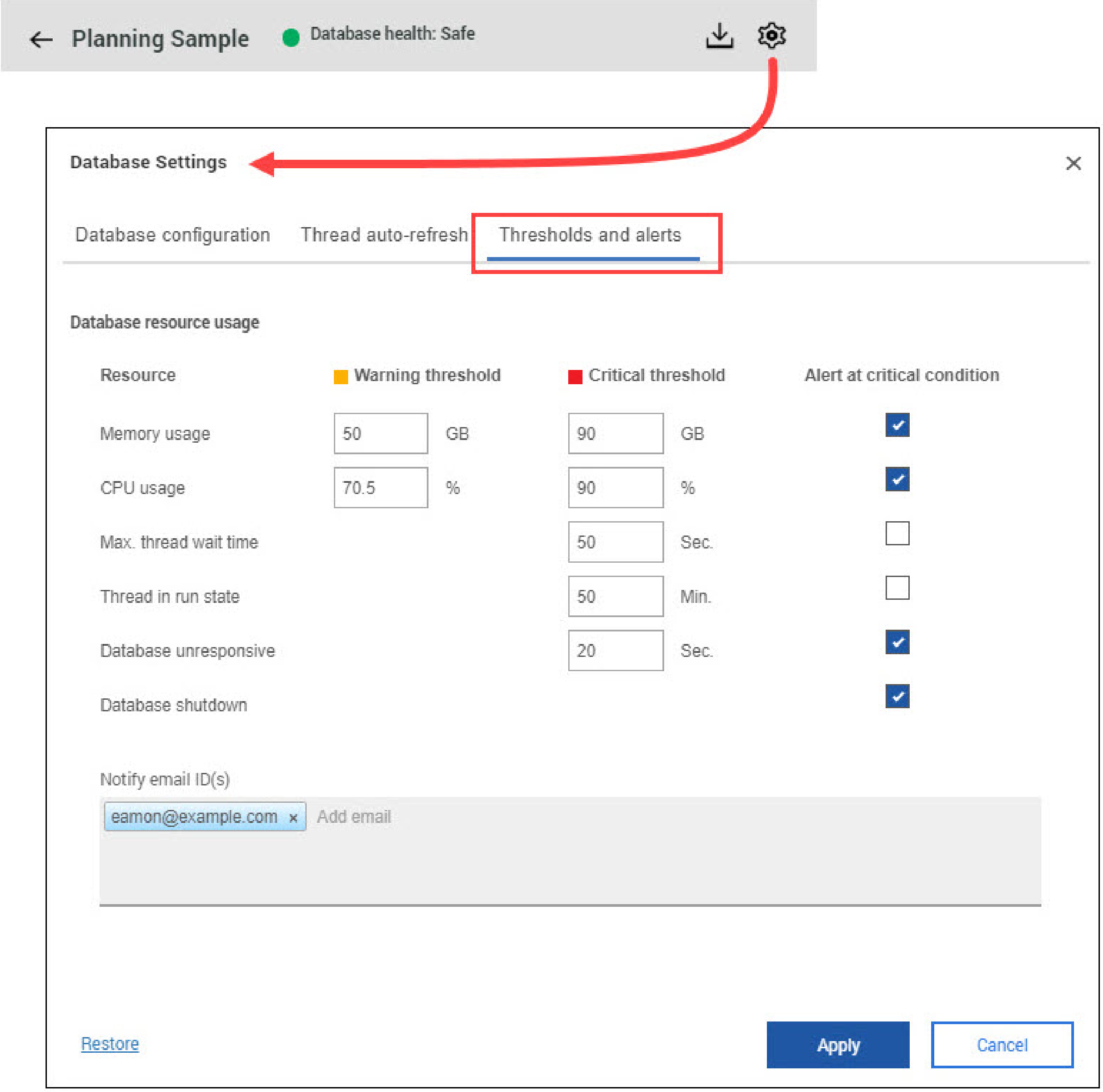
To learn more, see Visualization (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/c_visualizations_in_paw.html).

See thresholds and alerts for individual databases from the database activity report page now

Database threshold and alert configuration settings are now combined in a single tab on the **Database settings** page.

In previous versions of Planning Analytics Administration, thresholds and alerts were set on separate tabs of a configuration page and the settings applied to all databases in your environment. It was not possible to apply unique settings per database.

Database threshold and alert configuration settings are now combined in a single tab on the **Database settings** page for an individual database in Planning Analytics Administration. This simplifies the configuration process and lets you apply unique threshold and alert settings for each database in your environment. You can now access the **Database settings** page by clicking the ⚙ icon on a database activity report.



Database resource usage

Resource	🟧 Warning threshold		🟥 Critical threshold		Alert at critical condition
Memory usage	50	GB	90	GB	☑
CPU usage	70.5	%	90	%	☑
Max. thread wait time			50	Sec.	□
Thread in run state			50	Min.	□
Database unresponsive			20	Sec.	☑
Database shutdown					☑

Notify email ID(s)

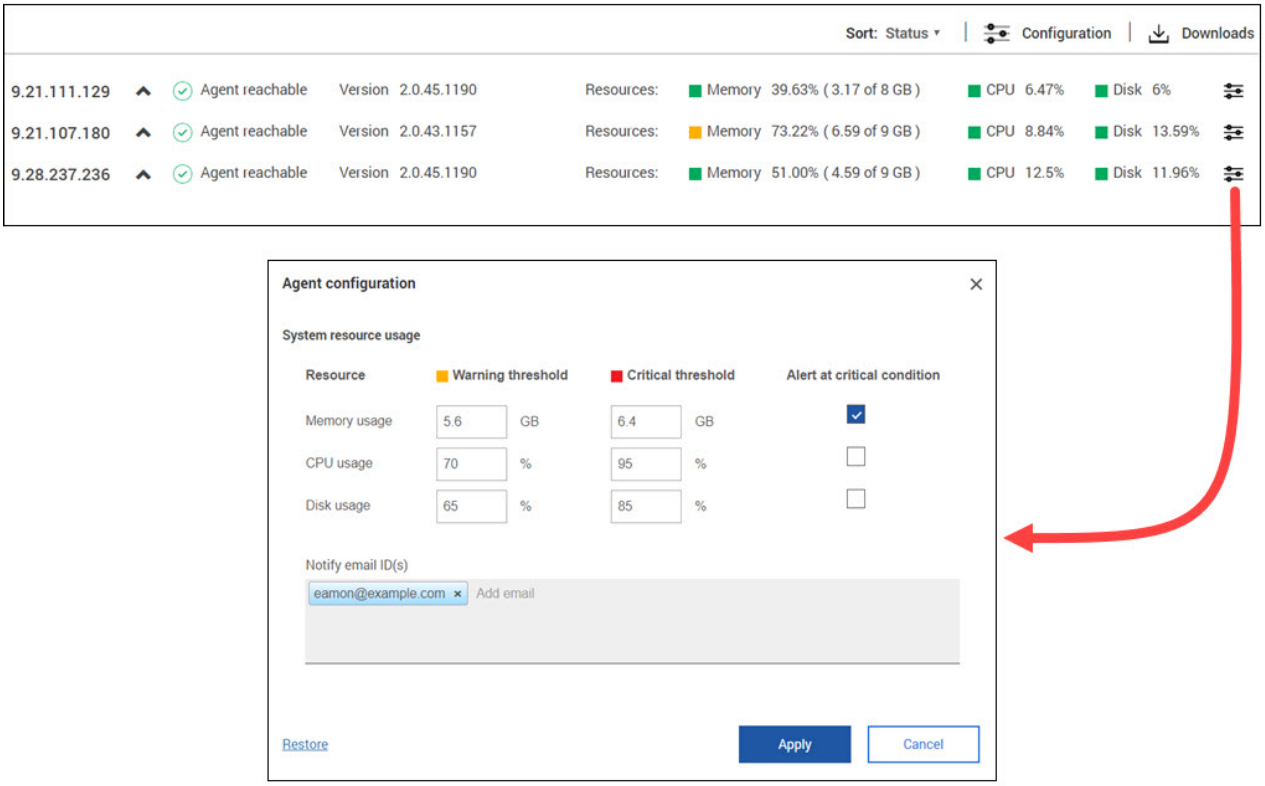
eamon@example.com × Add email

To learn more, see Set database thresholds and alerts (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_set_database_thresholds_and_alerts.html).

View system resource thresholds and alerts combined in a single configuration page now

You can now configure system resource thresholds and alerts on a single page in Planning Analytics Administration.

Configuring system resource thresholds and alerts on a single page simplifies configuration in all environments, both cloud and local. Additionally, this change provides greater control when you use Planning Analytics Administration Local to monitor multiple agents, as you can configure unique settings for each agent. The following image shows multiple agents in a Planning Analytics Administration Local environment, but the feature is identical in Planning Analytics Administration on Cloud.



9.21.111.129	^	✓ Agent reachable	Version 2.0.45.1190	Resources:	🟩 Memory 39.63% (3.17 of 8 GB)	🟩 CPU 6.47%	🟩 Disk 6%	☰
9.21.107.180	^	✓ Agent reachable	Version 2.0.43.1157	Resources:	🟧 Memory 73.22% (6.59 of 9 GB)	🟩 CPU 8.84%	🟩 Disk 13.59%	☰
9.28.237.236	^	✓ Agent reachable	Version 2.0.45.1190	Resources:	🟩 Memory 51.00% (4.59 of 9 GB)	🟩 CPU 12.5%	🟩 Disk 11.96%	☰

Agent configuration ×

System resource usage

Resource	🟧 Warning threshold	🟥 Critical threshold	Alert at critical condition
Memory usage	5.6 GB	6.4 GB	☑
CPU usage	70 %	95 %	☐
Disk usage	65 %	85 %	☐

Notify email ID(s)

eamon@example.com × Add email

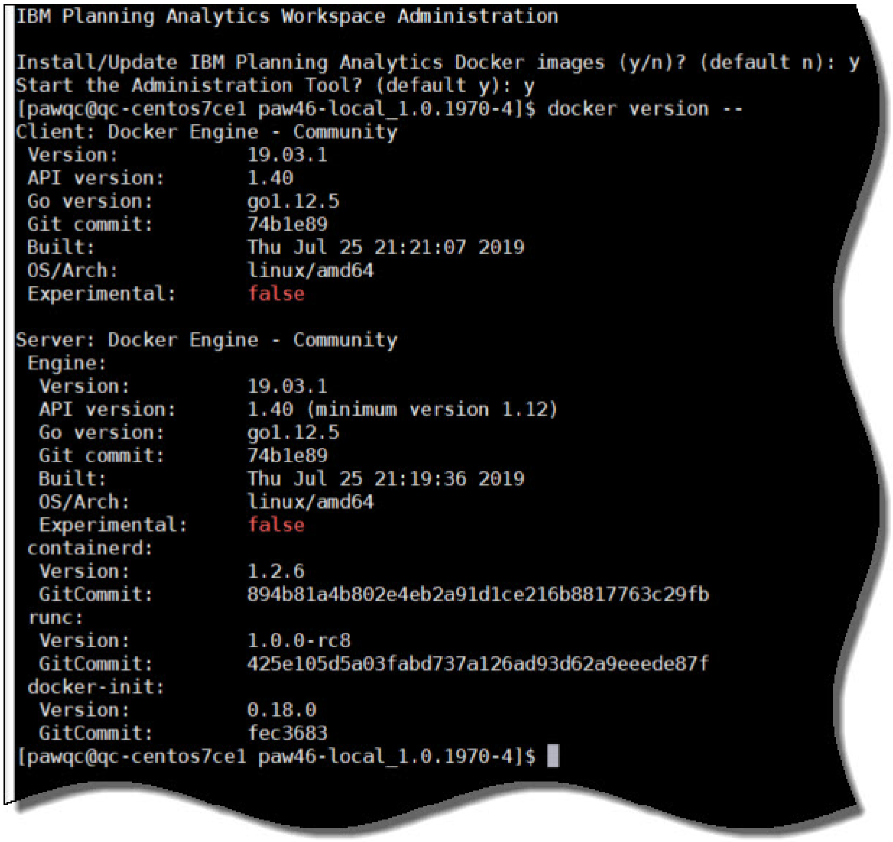
You can click ☰ next to the **Resources** status metrics to open the **System resource usage** configuration page.

In previous versions of Planning Analytics Administration, system resource thresholds and alerts were set on separate tabs of a configuration page and the settings applied to all agents in your environment. It was not possible to apply unique settings per agent.

To learn more, see Set system resource thresholds and alerts (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_set_system_resource_thresholds_and_alerts.html).

Install Planning Analytics Workspace Local on CentOS

⌂ Planning Analytics Workspace Local is now supported on Community Enterprise Operating System (CentOS). CentOS is a free, community-supported, open source Linux distribution. Planning Analytics Workspace Local on CentOS needs Docker Community Edition.



```
IBM Planning Analytics Workspace Administration

Install/Update IBM Planning Analytics Docker images (y/n)? (default n): y
Start the Administration Tool? (default y): y
[pawqc@qc-centos7ce1 paw46-local_1.0.1970-4]$ docker version --
Client: Docker Engine - Community
 Version:           19.03.1
 API version:       1.40
 Go version:        go1.12.5
 Git commit:        74b1e89
 Built:             Thu Jul 25 21:21:07 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.1
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.5
  Git commit:       74b1e89
  Built:            Thu Jul 25 21:19:36 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.6
  GitCommit:        894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc:
  Version:          1.0.0-rc8
  GitCommit:        425e105d5a03fabd737a126ad93d62a9eeede87f
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
[pawqc@qc-centos7ce1 paw46-local_1.0.1970-4]$ 
```

To learn more, see Installing Planning Analytics Workspace Local (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_install_overview.html).

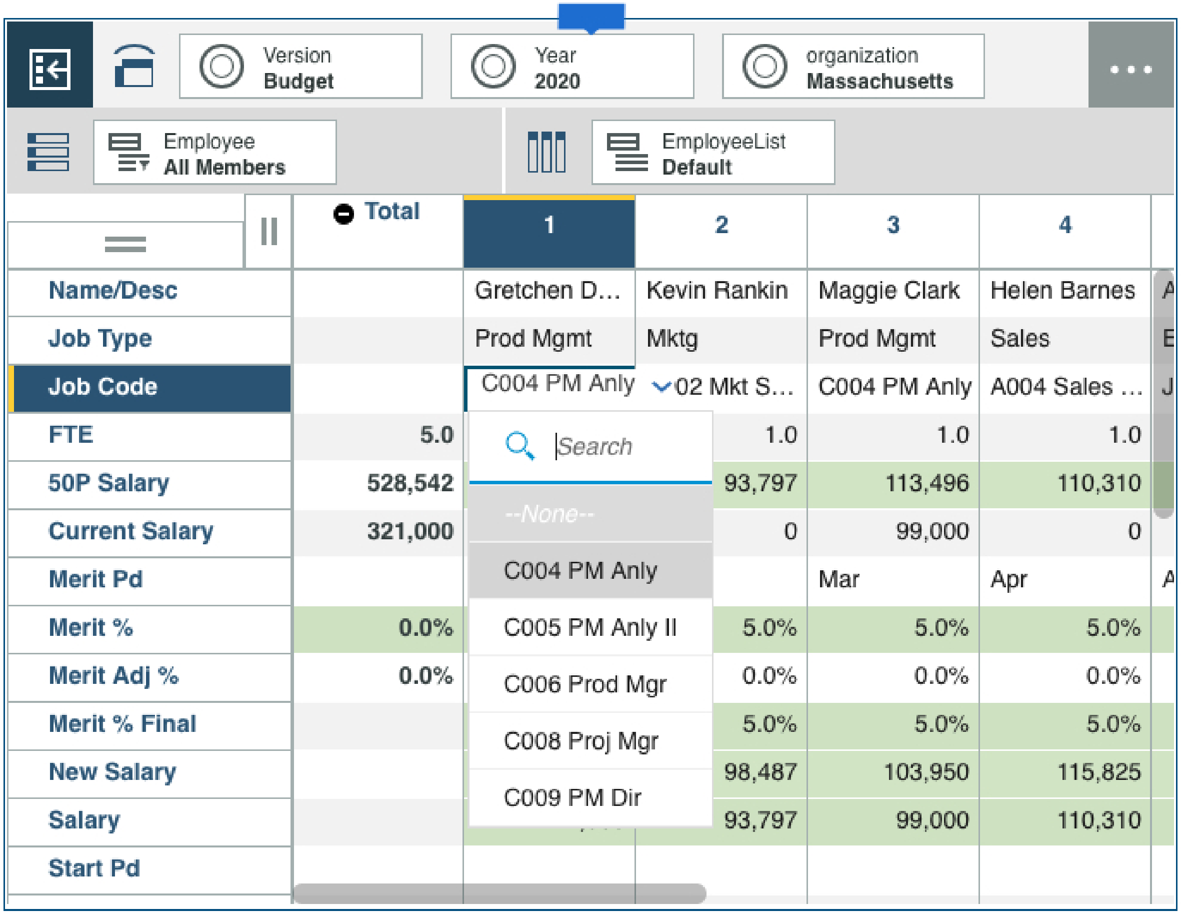
2.0.45 - What's new, August 21, 2019

See what's new in version 2.0.45 of IBM Planning Analytics Workspace in the following topics.

Greater control over pick lists

You now have greater control over which cube cells contain pick lists, by creating a pick list control cube.

Pick list control cubes give you both precision and flexibility by enabling you to define which individual cells a pick list is available from. You can also create rules for the pick list control cube, which allows you to define pick lists for any section of a cube, from a single cell to the entire cube.



Version Budget | Year 2020 | organization Massachusetts

Employee All Members | EmployeeList Default

	Total	1	2	3	4
Name/Desc		Gretchen D…	Kevin Rankin	Maggie Clark	Helen Barnes
Job Type		Prod Mgmt	Mktg	Prod Mgmt	Sales
Job Code		C004 PM Anly	02 Mkt S…	C004 PM Anly	A004 Sales …
FTE	5.0		1.0	1.0	1.0
50P Salary	528,542		93,797	113,496	110,310
Current Salary	321,000		0	99,000	0
Merit Pd				Mar	Apr
Merit %	0.0%		5.0%	5.0%	5.0%
Merit Adj %	0.0%		0.0%	0.0%	0.0%
Merit % Final			5.0%	5.0%	5.0%
New Salary			98,487	103,950	115,825
Salary			93,797	99,000	110,310
Start Pd					

Search

--None--

C004 PM Anly

C005 PM Anly II

C006 Prod Mgr

C008 Proj Mgr

C009 PM Dir

To learn more, see Create a pick list control cube (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_pick_list_control.html).

Give your dashboards a corporate look and feel using custom fonts and chart palettes

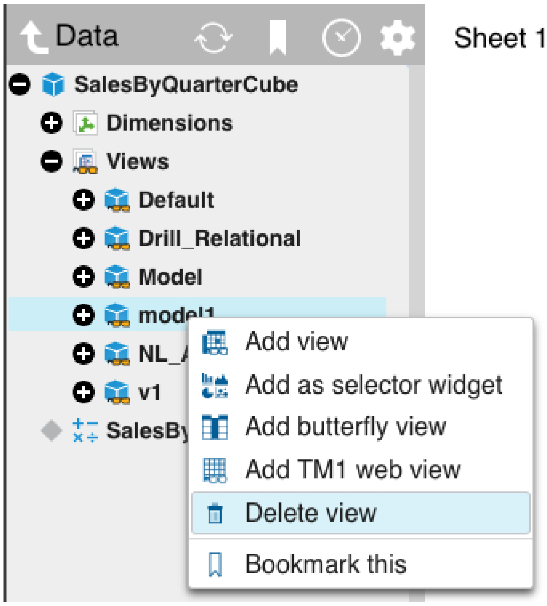
You can customize your dashboards with corporate fonts, and use corporate color palettes for charts, allowing you to give dashboards your corporate identity.

Administrators load the chart palettes and fonts in **Settings**.

To learn more, see Upload custom fonts (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_customize_fonts.html) and Add or remove color palettes for charts (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_customize_chart_patettes.html).

Delete a view from the content tree

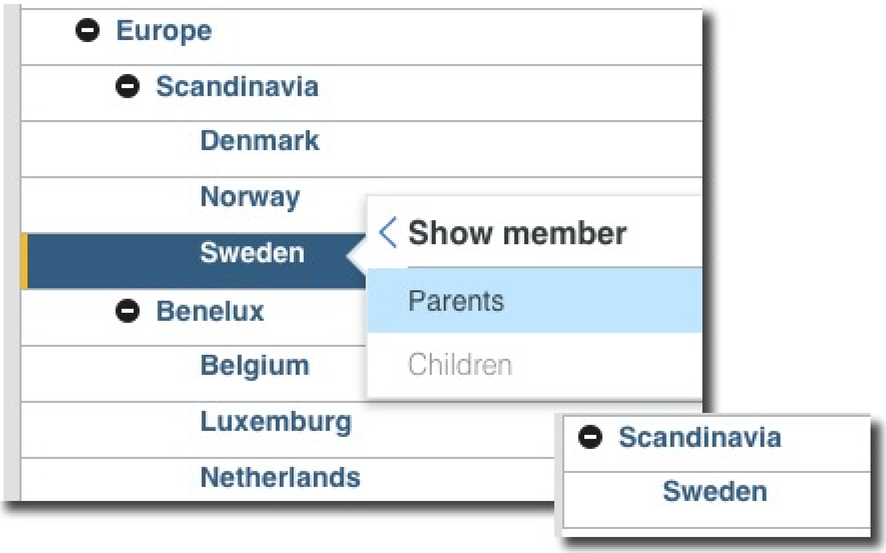
You can now delete a database view directly from the content tree in IBM Planning Analytics Workspace if you are logged on as a modeler.



To learn more, see Delete a cube or a view (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_delete_cube.html).

Show parents and children of members in the dimension editor

You can show the immediate parent of selected members and show the children.

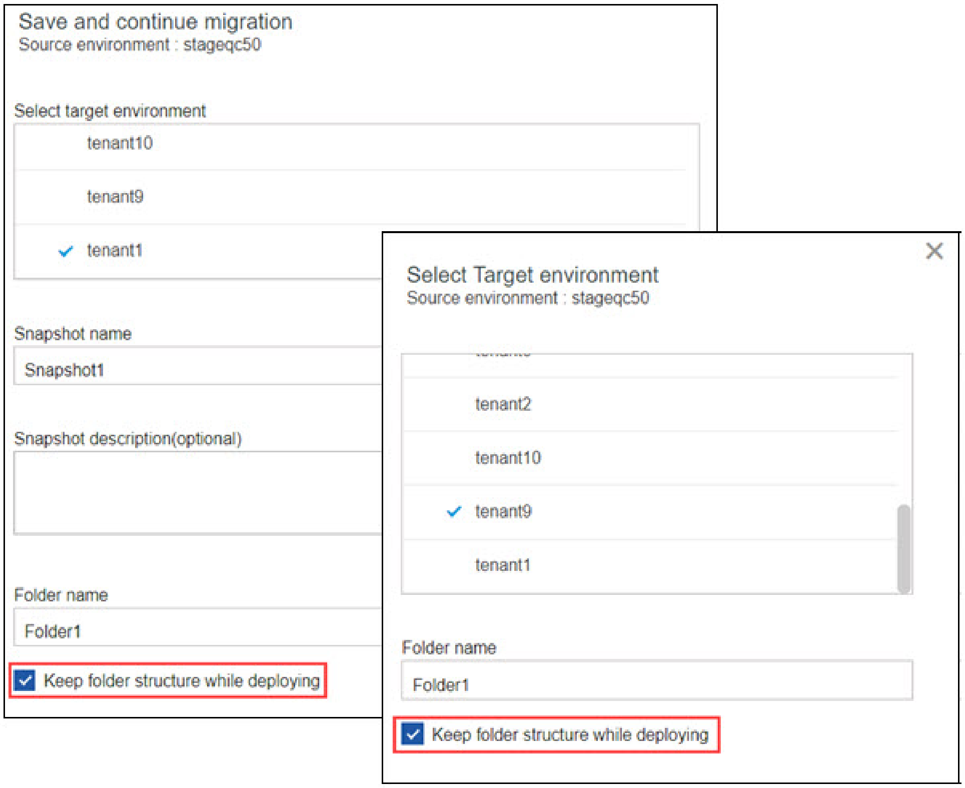


To learn more, see Add members to a dimension (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/t_paw_add_members_to_dimension.html).

Retain your folder structure when migrating assets

When you migrate a snapshot in Planning Analytics Administration, you can now choose to retain the folder structure of your assets.

When you enable the **Keep folder structure while deploying** option, the folder structure from your source environment is automatically created on your target environment. You no longer need to manually replicate the folder structure from your source environment on your target environment when migrating assets!



Save and continue migration
Source environment : stageqc50

Select target environment

- tenant10
- tenant9
- ✓ tenant1

Snapshot name: Snapshot1

Snapshot description(optional)

Folder name: Folder1

☑ Keep folder structure while deploying

Select Target environment
Source environment : stageqc50

- tenant2
- tenant10
- ✓ tenant9
- tenant1

Folder name: Folder1

☑ Keep folder structure while deploying

To learn more, see Get started with Lifecycle Management (https://www.ibm.com/support/knowledgecenter/en/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/paw_asset_workflow.html).

Chapter 2. Get started with Planning Analytics Workspace

The IBM Planning Analytics Workspace software is accessed in the cloud and as part of the IBM Planning Analytics Local software on premises.

For Planning Analytics Workspace in the cloud, log in with the credentials that are provided by your administrator.

To install Planning Analytics Workspace locally, see "Planning Analytics Workspace installation" in the *Planning Analytics Local Installation and Configuration* documentation on IBM Knowledge Center.

User roles

The following roles are available in Planning Analytics Workspace. The default role is **Analyst**.

Consumer

Consumers can open books and views and other content that is shared with them.

Consumers cannot create their own books and views, but they can share content that is shared with them, with consumer rights only.

Consumers can delete books and views if they have **Full control** permission set for those books and views.

Note: When you log in to Planning Analytics Workspace from an iPad, you are always a Consumer, regardless of your actual role. For more information, see "Accessing Planning Analytics Workspace from Apple iPad" on page 36.

Analyst

Analysts have the rights of a consumer, plus the following abilities:

Analysts can create, edit, and share books and views.

Analysts can delete books and views if they have **Full control** permission set for those books and views.

Modeler

Modelers have the rights of an analyst plus the following abilities:

They can create and share content, and edit content that is shared with them.

Modelers can create, edit, and delete dimensions and hierarchies. They can add, delete, cut, paste, copy, move, and sort members and their attributes in a hierarchy.

For more information, see "The modeler role" on page 184.

Administrator

Administrators have all the rights of a modeler, plus the following abilities:

Administrators can see all content in the workspace.

Administrators can set permissions for a book.

Note: At least one user in your organization must have the administrator role. If your organization has only one administrator, this user cannot be deleted or assigned to another role. Administrators can assign roles to users and only an administrator can change another user's role to administrator.

By default, the administrator role is assigned to the first user in your organization's Planning Analytics Workspace account. In Planning Analytics Workspace on cloud, this first administrator is also known as the *subscription administrator*. Only the subscription administrator can add new users.

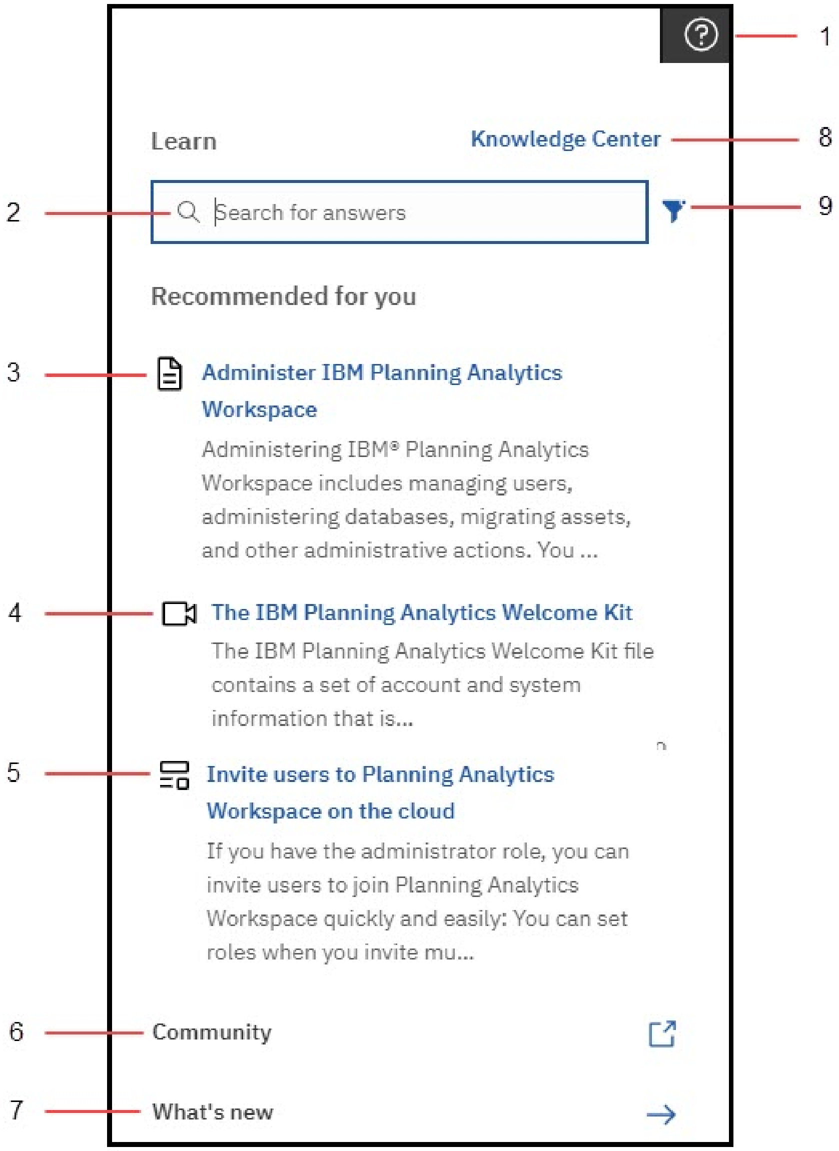
Find information

Whether you are using Planning Analytics Workspace or Planning Analytics Workspace Classic, you can view the complete set of Planning Analytics documentation on IBM Knowledge Center (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0). Keep reading for details on the unique integrated help systems in Planning Analytics Workspace and Planning Analytics Workspace Classic.

Find information in Planning Analytics Workspace

If you want to learn more about how to use Planning Analytics Workspace, click the Help icon ⓘ and ask a question in the cognitive **Learn pane**. The **Learn pane** tailors your help experience based on where you are working in the product and finds only the answers that are relevant to your user role. You can find the latest videos, blogs, and documentation.

Use these features on the **Learn pane** to find answers and be more productive!


1
Learn
Knowledge Center
8
2
Search for answers
9
Recommended for you
3
Administer IBM Planning Analytics Workspace

Administering IBM® Planning Analytics Workspace includes managing users, administering databases, migrating assets, and other administrative actions. You ...

4
The IBM Planning Analytics Welcome Kit

The IBM Planning Analytics Welcome Kit file contains a set of account and system information that is...

5
Invite users to Planning Analytics Workspace on the cloud

If you have the administrator role, you can invite users to join Planning Analytics Workspace quickly and easily: You can set roles when you invite mu...

1

Click the Help icon to open the **Learn pane** (it remembers where you were the last time you opened it). Click anywhere to close it. The **Learn pane** recommends content that relates to your task and finds similar content that you might also like. And, it is always learning! When you search and find answers, you are training the **Learn pane** and those answers influence future recommendations.

2

Type a question in the Search box. You can search in any supported language in the Learn pane and you see translated documentation in your search results. You also see blogs and videos that match your search, however, blogs and videos aren't translated.

3

Read the formal product documentation, sourced from IBM Knowledge Center.

4

Watch a video! Sometimes the best way to learn is to see it in action.

5

Read a post in the Planning Analytics Community Blog The community blog posts are written by experts who use Planning Analytics Workspace and share their tips and tricks.

6

Go to the Planning Analytics Community. In the community, you can find the latest articles, blog posts, and events. You can also start and contribute to discussions about Planning Analytics.

7

Click **What's New** to find out what is new in the latest release of Planning Analytics Workspace.

8

Visit the IBM Knowledge Center for all IBM Planning Analytics documentation In the IBM Knowledge Center, you can read all documentation, including related products.

9

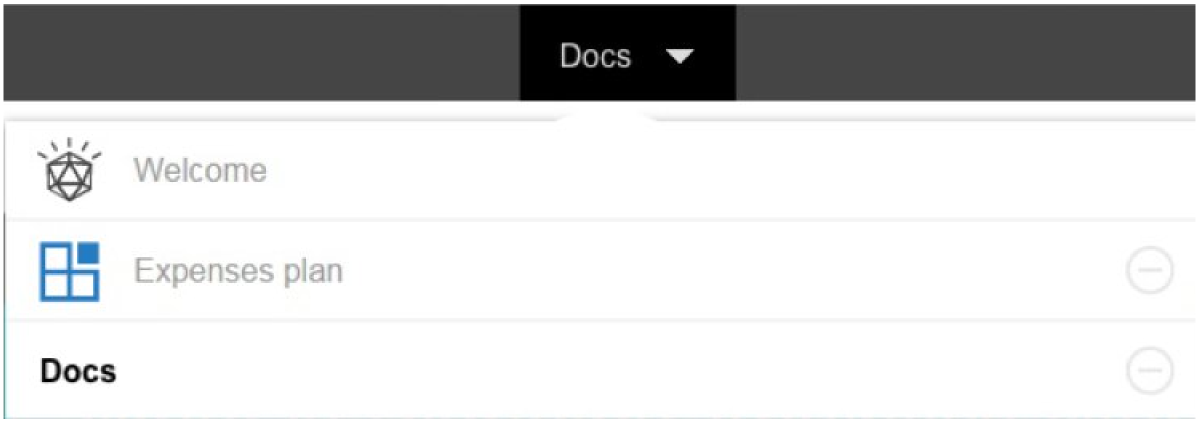
Filter your search results to show only your preferred content type: videos, blogs, or documentation.

Find information in Planning Analytics Workspace Classic

You can access the integrated help system from within the Planning Analytics Workspace Classic application. To view the help, click ?, and then click **Docs**.

Note: If you have accessibility requirements, you can view the documentation on IBM Knowledge Center. Click ? > **Docs**, and then click **Accessible Docs**.

To navigate back to Planning Analytics Workspace, click **Docs**:

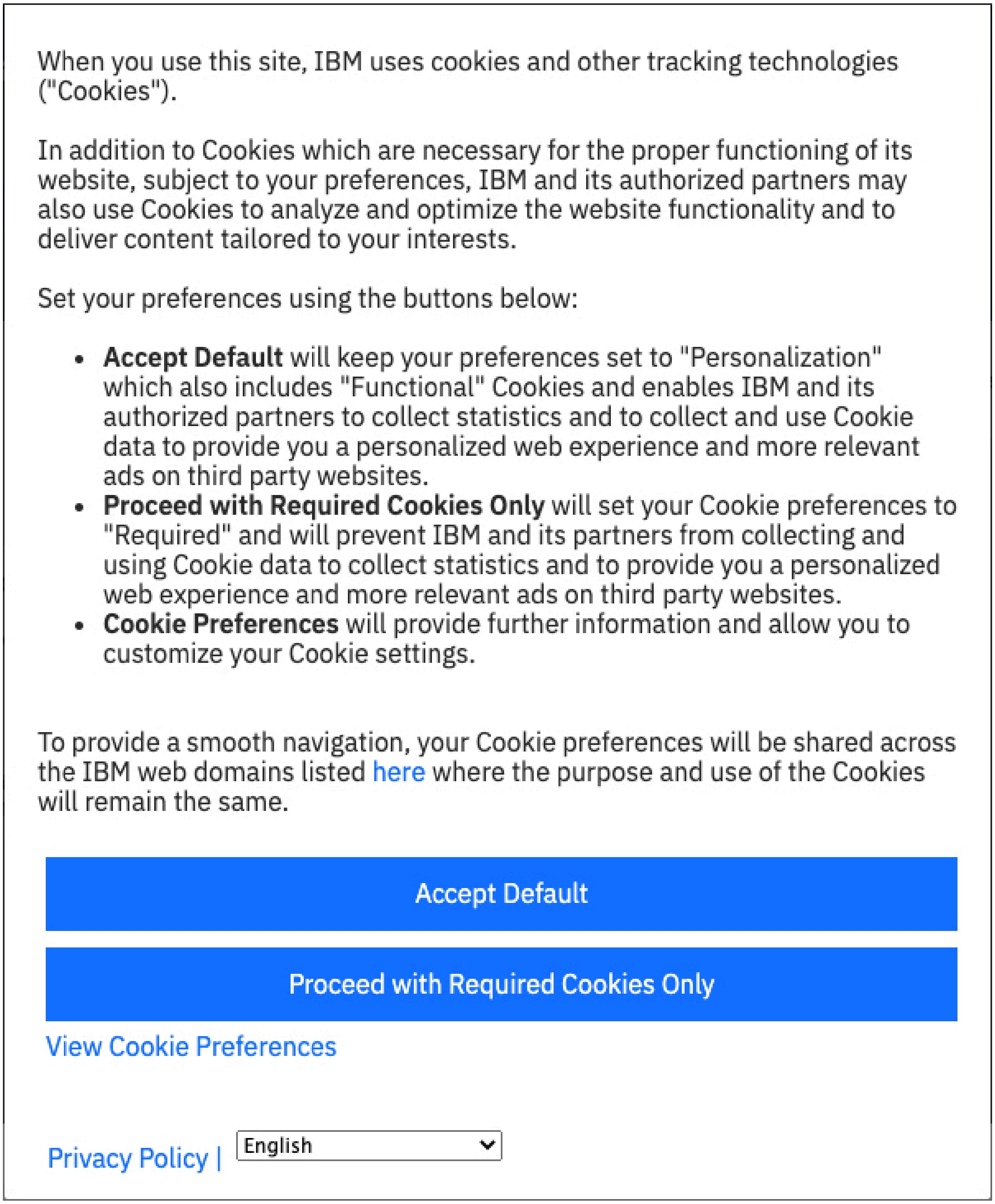


Then, click either **Welcome** or the name of the book.

Cookie preferences for Planning Analytics on cloud

IBM Planning Analytics creates cookies when you go to the website, and saves them to your computer. You can customize the level of cookies that are created.

If you are based in the European Union, you will see a prompt the first time that you log on after Planning Analytics Workspace on cloud is upgraded. Users are determined to be in the European Union based on their IP address by using geolocation services. Your acceptance of cookie preferences is stored as a cookie in your browser, so if you clear your cookies or change browser, you see the prompt again. This is standard behavior, and enables you to control the level of cookies that you allow to be stored.



When you use this site, IBM uses cookies and other tracking technologies ("Cookies").

In addition to Cookies which are necessary for the proper functioning of its website, subject to your preferences, IBM and its authorized partners may also use Cookies to analyze and optimize the website functionality and to deliver content tailored to your interests.

Set your preferences using the buttons below:

- **Accept Default** will keep your preferences set to "Personalization" which also includes "Functional" Cookies and enables IBM and its authorized partners to collect statistics and to collect and use Cookie data to provide you a personalized web experience and more relevant ads on third party websites.
- **Proceed with Required Cookies Only** will set your Cookie preferences to "Required" and will prevent IBM and its partners from collecting and using Cookie data to collect statistics and to provide you a personalized web experience and more relevant ads on third party websites.
- **Cookie Preferences** will provide further information and allow you to customize your Cookie settings.

To provide a smooth navigation, your Cookie preferences will be shared across the IBM web domains listed here where the purpose and use of the Cookies will remain the same.

If you are not based in the European Union, you are not prompted for cookies but you can select the level of cookies by clicking 👤 and then selecting **Cookie Preferences**. You can see more information about the levels of cookies by clicking **View Cookie Preferences** in the **Cookies** window.

The following options are available.

Required Cookies
The minimum level of cookies needed to use IBM Planning Analytics.These cookies are needed to provide basic functions as you browse IBM websites.

Required Cookies enables features such as:

- Cookie consent collection and cookie blocking for privacy controls.
- Secure log-in and transactions (that is, authentication, single sign-in, remember settings) across web pages.
- Online services, load balancing, and performance.
- Remembering transaction progress during the session.

Functional Cookies

Functional Cookies provides a better user experience than Required Cookies. The Functional Cookies selection captures and remembers user preferences in IBM websites, enhances their usability, analyzes site usage, and enables site optimization.

In addition to the **Required Cookies** features, **Functional Cookies** enables functions such as:

- Remembering your log-in details for automatic log-in across session.
- Analyzing site usage to provide better content experiences.
- Conducting analytics to optimize site functions.
- Making sure that the website looks and feels consistent.

Personalization Cookies

Personalization Cookies improve your overall experience of IBM websites and tailor content/ advertising to your interests.

- Allow personalization of IBM page content.
- Allow delivery of IBM marketing relevant to your interests on IBM and on third-party publishers.

Applicability of Planning Analytics Workspace documentation

The documentation available through the integrated help system and on IBM Knowledge Center always reflects the most recent full capabilities of Planning Analytics Workspace on cloud.

Cloud versus on-premises

In some instances, the current capabilities on cloud differ from the most recent Planning Analytics Workspace Local update. This is because local updates are made available only after cloud deployment occurs. In this case, Planning Analytics Workspace Local customers might see documentation for features that are not yet available to them.

Similarly, while Planning Analytics Workspace on cloud is refreshed on a regular cadence, Planning Analytics Workspace Local customers might choose to update their installation on a less frequent basis. This can also result in a difference between the features that are described in the documentation and the features that are available to a Planning Analytics Workspace Local customer.

Planning Analytics Workspace versus Planning Analytics Workspace Classic

A new user experience was introduced in Planning Analytics Workspace 2.0.57 SC. This new experience includes an updated look and feel, as well as several significant new features.

While the new experience is available to all customers, it is not mandatory, and some customers might choose to continue using the 'classic' user interface. When there is a divergence between the procedures required to perform a task in the new experience and the classic interface, the documentation describes both interfaces. The new experience interface introduced in 2.0.57 SC is identified as Planning Analytics Workspace in the documentation, while the older interface is identified as Planning Analytics Workspace Classic.

For an example of how the two user interfaces are differentiated in the documentation, see Monitor and manage database threads.

Deprecation

In some cases, the current documentation differs from what is available in your environment because functionality is deprecated in the current release of IBM Planning Analytics. In the documentation, Deprecated indicates deprecated functionality that will be removed in a future release. Not supported indicates functionality that is no longer available or supported in the current release.

TM1 Server

Depending on the Planning Analytics version that a Planning Analytics Workspace cloud customer uses, some features that are documented might not be available to use or might not be visible in the user interface.

Device and browser compatibility

Some features that are documented and available in Planning Analytics Workspace on a desktop browser might not be available on Apple iPad. For more information, see "Accessing Planning Analytics Workspace from Apple iPad" on page 36.

Accessing Planning Analytics Workspace from Apple iPad

You can access Planning Analytics Workspace from an Apple iPad. Other mobile devices are not currently supported in Planning Analytics Workspace.

You can view this video to learn how to build books for using on Apple iPad in Planning Analytics Workspace.

https://youtu.be/BaUjhYw8IK4

When you log in to Planning Analytics Workspace from an iPad, you are always working in Consumer mode, regardless of your actual defined role assignment.

In Consumer mode you can:

- Open the Welcome page and view the folder structure
- Open existing views, books, and websheets
- Modify cell values
- Switch visualization types
- Interact with explorations and visualization (drill down, hide members, reorder dimension, and so on)
- Use selectors to set context for explorations and visualizations
- Participate in collaborative chats

In Consumer mode on a mobile device, you *cannot*:

- View or access the data tree
- Use or enable Edit mode
- Perform any administration tasks
- Save an existing view or book, or use save-as to create a copy of a view or book
- Create a view or book
- Open the Set Editor to modify existing sets or create new sets
- Create member calculations or summary calculations
- Share a book or view in any manner (download, email, or print)
- Delete or rename a book or view
- Add a view to a collection

In addition, you cannot create conditional formatting on an iPad.

Performance considerations

When you open a view on an iPad, Planning Analytics Workspace initially loads and displays only the number of rows that fit on a single page. Depending on how the view is configured, this can be anywhere between 30 - 50 rows.

As you scroll through the view, Planning Analytics Workspace loads each subsequent page sequentially. This occurs even if you rapidly scroll through several pages without pausing for each page to be displayed. For example, if you are looking at a value on the first page of a view, and then scroll to page 8 of the view, pages 2 - 7 must be loaded before page 8 can be displayed. There is a slight delay as each page is loaded.

To maintain minimal load times, limit views that are accessed from iPads to 120 or fewer rows.

Additionally, you can improve performance by:

- Limiting the number of views on a single sheet (the best performance is achieved with a single view on a sheet)
- Using chart visualizations in place of explorations (grids) whenever practical
- Limiting the scope of synchronization between views to the lowest possible level

Supported languages

IBM Planning Analytics Workspace supports different languages in its user interface as well as its documentation.

The following table shows the languages supported in the user interface and in the documentation.

Table 1. Supported languages in Planning Analytics Workspace.

Language	User interface	Documentation
Brazilian Portuguese	✓	✓
Danish	✓	✓
English	✓	✓
French	✓	✓
German	✓	✓
Italian	✓	✗
Japanese	✓	✓
Norwegian (Bokmål)	✓	✓
Polish	✓	✓
Russian	✓	✓
Simplified Chinese	✓	✓
Spanish	✓	✓
Swedish	✓	✓
Traditional Chinese	✓	✓

Chapter 3. Work in books and views

Books contain related data in the form of views, websheets, scorecards, visualizations, graphics, videos, and embedded web pages. Views are defined selections of a cube that are used for analysis, exploration, and data entry.

Books

Books can contain cube views, charts, scorecards, dimension selectors, graphics, videos, and embedded links to web pages.

You can do the following tasks in a book:

- Collect and arrange content to support a process.
- Add navigation and text to guide users through a process.
- Customize the formatting and layout by adding logos, videos, graphics, and applying formats and text styles.
- Include content from different cubes and databases in the same book.
- Share books and views with people.

Create a book

Create an IBM Planning Analytics Workspace book to contain your data. Use a defined template or build your workspace with a freeform template.

Create a book in Planning Analytics Workspace

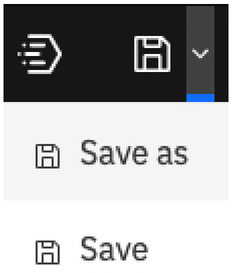
Procedure

1. In a web browser, go to the URL for IBM Planning Analytics Workspace and log in with your credentials.
2. Click **New** +, then click **Book from template**.
3. Select a template to use and click **OK**.

   IBM Planning Analytics Workspace provides templates that contain predefined designs and grid lines for easy arrangement and alignment of the visualizations.

   When you create a new book, choose a tabbed template to create a book with multiple tabs or choose a single page template. You can also hide tabs on a tabbed book if you want just one tab and less clutter on the screen when you switch to preview mode. For more information, see "Hiding the tabs " on page 46.
4. Alternatively, choose **Book** to create a book without having to specify a template or layout. This option is slightly faster, and when you choose this option, the new book is created with the blank tabbed template.
5. Click the **Properties** icon to modify book properties such as visual themes, layout positioning, and tab style options. To close the **Properties** pane, click the **Properties** icon again.

6. You can rename your book when you are in edit mode. Click the down arrow beside the **Save** icon then click **Save as**.

7. Save your book by clicking and selecting whether the book is **Shared** or **Personal**. If you have folders, you can select which folder the book is saved in.

Create a book in Planning Analytics Workspace Classic

Procedure

1. In a web browser, go to the URL for Planning Analytics Workspace and log in with your credentials.
2. Click **New** ⊕, then click **Book from template**.
3. Name your book with a meaningful name to make it easier to search for.
4. Select a workspace layout and template for the initial sheet in your book, then click **Create**.

   The default selection of a tabbed workspace with a freeform template is a good starting point. The choices of workspace layout are the following:

   Single page
   Displays all of your data on one sheet.

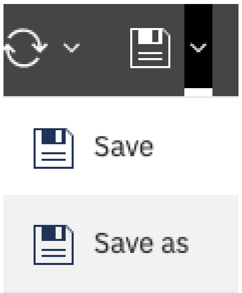
   Tabbed
   You can create separate sheets for different categories of information. For example, you can create separate tabs for sales by country, by region, and by total sales.

   You can choose from the following templates:

   Freeform
   The Freeform template is one large area that is not divided into sections. Objects in this template can be positioned wherever you want, and they keep their size and position regardless of the screen size.

   Other templates
   A number of defined templates that divide the sheet into separate sections are available. This kind of template uses relative positioning. Images adjust their size and position to fit the space they are added to, but views don't change.
5. Alternatively, choose **Book** to create a book without having to specify a template or layout. This option is slightly faster, and when you choose this option, the new book is created with the freeform layout and the tabbed workspace.
6. You can choose themes and style options for the book. Click the **Properties** icon . To close the **Properties** pane, click the **Properties** icon again.

7. You can rename your book when you are in edit mode. Click then click **Save as**.



8. Save your book by clicking and selecting whether the book is **Shared** or **Personal**. If you have folders, you can select which folder the book is saved in.

What to do next

Next, add some content to your book. You can add views, visualizations (charts), graphics, videos, embedded web pages, and text.

Edit mode

Switch to edit mode to modify and assemble a book.

Edit mode in Planning Analytics Workspace

When a book is in edit mode, you can do the following tasks:

- Add new content.
- Move and resize content.
- Make sure that any changes that you make to widget settings are kept. For example, if you are not in edit mode, some actions such as pivoting and slicing are not kept across sessions.
- Rename the book.
- Use properties to configure colors, themes, layouts, and so on.
- Add new tabs to create a book with multiple tabs and to organize your data. Click .

Click to go into edit mode, and to leave edit mode.

Edit mode in Planning Analytics Workspace Classic

When a book is in edit mode, you can do the following tasks:

- Add new content.
- Move and resize content.
- Make sure that any changes that you make to widget settings are kept. For example, if you are not in edit mode, some actions such as pivoting and slicing are not kept across sessions.
- Rename the book.
- Add new dashboards to organize your data. Click .

Click to go into edit mode, and to leave edit mode.

Setting up the page in a book

You can set up your book to create the layout you need for your business purposes.

About this task

You can set whether the layout positioning is relative or absolute.

When you choose a layout style, consider the screen size of the device that will be used to view the book.

You can define the page size or select the page size from a number of presets.

If you choose absolute layout positioning, the objects retain their size and position, regardless of the screen size of the device that they are being viewed in. For small screen sizes, absolute positioning can increase the need for scrolling because objects are larger than the screen. Absolute positioning results in one large area that is not divided into sections. Choose absolute if you want visualizations to appear exactly as you size and place them, regardless of the screen size of the device that they are viewed in.

The option **Fit page** resets the view of the book to fit your screen. When you select absolute layout positioning, then the option **Fit page** is not available.

If you choose relative layout positioning, the objects adjust their size and position relative to each other, the data they show, and the screen size of the device that they are viewed in. Select relative layout positioning if you want the size and position of visualizations to adjust to fit into the screen size of the device that they are viewed in. The appearance of the book can change depending on the size of the screen that it is viewed in.

Procedure

1. Open a book.
2. Click the **Properties** icon.

   If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. Under **Canvas**, select a relative or absolute layout positioning.
4. Under **Page size** select the page size from the **Preset** menu or specify the width and height yourself. Optionally select the **Fit page** option.

Snapping objects to a grid and to each other

You can display a grid on your book to help you place objects exactly where you want them. You can snap objects to the grid and snap objects to each other.

Procedure

1. Click the **Properties** icon.
2. If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. For the entire book, ensure that nothing else has focus by clicking the background, away from any visualization or object.
4. On the **General** tab, open **Canvas**.
5. In the **Grid** section, complete one or more of the following actions:

Goal	Action
Show or hide the grid on the canvas	Set the **Show grid** option to on or off. This option is on by default.

Goal	Action
Snap objects to the grid on the canvas	Set the **Snap to grid** option to on or off. This option is on by default.
Snap objects to each other	Set the **Snap to objects** option to on or off. This option is on by default.

Changing colors
You can change colors in the entire book, in a visualization, or in widgets that you added to a book.

- For the book, you can change the visual theme, the color palette, and the background color.
- For visualizations, you can change the color palette, the color used in the elements (such as bars, bubbles, or lines), the fill color, and the border color. You can also make the visualizations more transparent or opaque.
- For shape and text widgets, you can make the widget more transparent or opaque, change the fill color, and change the border color.

Procedure

1. Click the **Properties** icon.

   If you don't see the **Properties** icon, Click the **Edit or preview** icon.

2. For the entire book, ensure that nothing else has focus by clicking the background, away from any visualization or object and complete one or more of the following actions in the **General** tab:

What to change	Actions
The theme of the entire book	Click **Color and theme**, and under **Visual theme** select a different theme.
The background color of the entire book	Click **Color and theme**, and under **Background color** select a new color. Optionally, click **Select custom color** and use the color picker to create your own background color.

3. For visualizations, click the visualization that you want to work with and complete one or more of the following actions:

What to change	Actions
The color palette used in a visualization	Click the **Visualization** tab, and under **Color** click **Change color palette**. Select a new palette. Optionally, click **Heat palette** to change the heat scale order.
The color of the elements of a visualization (such as bars, bubbles, or lines) if the visualization does not use the **Color** data slot	Click the **Visualization** tab, and under **Color** click **Element color** and select a new color.
How transparent or opaque a visualization is	In the **General** tab, move the **Opacity** slider.
The fill color for a visualization	In the **General** tab, click **Fill color** and select a new color. Optionally, click **Select custom color** and use the color picker to create your own fill color.

What to change	Actions
The border color for a visualization	In the **General** tab, click **Border color** and select a new color. Optionally, click **Select custom color** and use the color picker to create your own border color.

What to change	Actions
The color palette used in a visualization	In the **Visualization** tab, click **Color** > **Change color palette** and select a new palette. Optionally, click **Change heat palette** to change the heat scale order.
The color of the elements of a visualization (such as bars, bubbles, or lines) if the visualization does not use the **Color** data slot	In the **Visualizations** tab, click **Color** > click **Element color** and select a new color.
How transparent or opaque a visualization is	In the **General** tab, move the **Opacity** slider.
The fill color for a visualization	In the **General** tab, click **Fill color** and select a new color. Optionally, click **Select custom color** to create your own color with the **Color picker**.
The border color for a visualization	In the **General** tab, click **Border color** and select a new color. Optionally, click **Select custom color** to create your own color with the **Color picker**.

4. For shape and text widgets that you added to the book, select the widget that you want to work with and complete one or more of the following actions in the **General** > **Appearance** tab:

What to change	Actions
How transparent or opaque a shape or text is	Move the **Opacity** slider.
The fill color for a shape or text	Click **Fill color** and select a new color. Optionally, click **Select custom color** to create your own color with the **Color picker**.
The border color for a shape or text	Click **Border color** and select a new color. Optionally, click **Select custom color** to create your own color with the **Color picker**.

Customizing tabs

Customize tab styling for individual tabs or for all the tabs in your book. You can move the tabs to the top, left, bottom, or right of your book and you can add icons to your tabs.

Customizing tab styling for individual tabs

For individual tabs on your book, you can change the tab title color, the bar under the title that indicates which tab is selected, and the tab fill color.

Procedure

1. Open a book that has at least one tab.
2. Click the **Properties** icon.

   If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. Click **Tabs**.

4. Under **Individual tab styling**, select the tab that you want to customize.
5. Select a color for the tab title.
6. Select a color for the bar under the tab title that indicates which tab is selected.
7. Select a fill color for this tab.

   Tip: You can also click **Select custom color** to open the color picker instead of choosing from the default color options.

Customizing tab styling for all tabs

At a global level for all tabs on your book, you can change the tab title color, the bar under the title that indicates which tab is selected, and the tab fill color.

Procedure

1. Open a book that has at least one tab.
2. Click the **Properties** icon.

   If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. Click **Tabs**.
4. Under **Master tab styling**, select a color for tab titles.
5. Select a color for the bar under the tab title that indicates which tab is selected.
6. Select a fill color for the tabs.

   Tip: You can also click **Select custom color** to open the color picker instead of choosing from the default color options.

Changing tab position

You can move the tabs to be displayed on the top, left, bottom, or right of your book.

Procedure

1. Open a book that has at least one tab.
2. Click the **Properties** icon.

   If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. Click **Tabs**.
4. From **Tab position**, select **Top**, **Left**, **Bottom**, or **Right**.

Adding icons to tabs

You can add icons to your tabs and control the placement and color of the icons.

Procedure

1. Open a book that has at least one tab.
2. Click the **Properties** icon.

   If you don't see the **Properties** icon, click the **Edit or preview** icon.
3. Click **Tabs**.

4. Under **Individual tab styling**, select the tab that you want to add an icon to.
5. Under **Visual**, select an icon.
6. Select a color for the icon.
7. Click **Back** on the properties pane.
8. Under **Tabs**, select the position of the icon on the tab, relative to the title, for example, **Right of title**.

Hiding the tabs

If your book has tabs, you might want the tabs hidden when the book is viewed in preview mode. You can hide the tabs to create more space on the screen.

Procedure

1. Open a tabbed book.
2. Click the **Properties** icon.
3. Click **Tabs**.
4. Click the **Show tabs in preview** toggle to hide the tabs.
5. Click the **Edit or preview** icon to see preview mode with the tab hidden.

Changing the template on a tabbed book

You can change the template while you're assembling a tabbed book. After you change the template, move the objects around to fit.

Procedure

1. On a book, select a tab and then click the selected tab.
2. Click the **Change template** icon.
3. Select a template.

Set dashboard properties

You can set the theme for a dashboard, the background color, whether to show or hide sheet titles, and the synchronization scope for a dashboard.

Procedure

In Edit mode, click . You can set the following properties:

a) In **Themes**, select the Default, Light, or Dark theme.

   You can customize the themes that are used in the dashboard. To learn more, see "Customize Planning Analytics Workspace" on page 340.

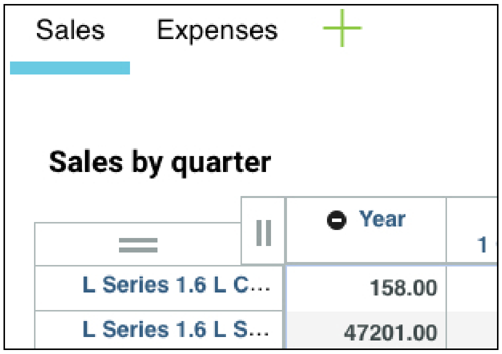
b) In **General style**, select the background color.

c) In **General style**, decide whether you want to see **Sheet Titles** on the dashboard. To hide sheet titles, clear the **Show to Viewers** option.

   Sheet titles are always visible when you are in Edit mode, so to see what the dashboard looks like, disable Edit mode. You can free up space on the dashboard by hiding sheet titles.

   Tip: If you hide the sheet titles, add navigation buttons to your view so that you can move to other sheets when the tabs aren't visible. To find out more, see "Buttons " on page 153.

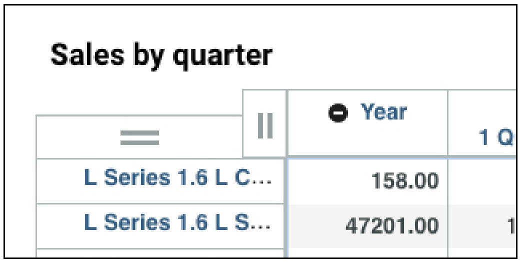
   The following image shows a view with sheet titles:



Sales by quarter

	Year	1
L Series 1.6 L C...	158.00	
L Series 1.6 L S...	47201.00	

The following image shows a view without sheet titles:


Sales by quarter

	Year	1 Q
L Series 1.6 L C...	158.00	
L Series 1.6 L S...	47201.00	1

d) In **Synchronization scope**, set the synchronization scope to either **Book** or **Sheet**.

To learn more, see "Synchronize objects in a book or sheet" on page 116.

Add pictures, videos, and web pages

You can add pictures, videos, and web pages to your Planning Analytics Workspace books. For example, you can add a company logo, and a video blog that explains results.

Before you begin

The objects that you add must be available from a URL. In a browser window, locate the objects, and copy the URL (address).

Note: You must have permission to use the content that you add, the content must be available from an address beginning with HTTPS, and the publisher of the content must allow it to be displayed in a frame.

Procedure

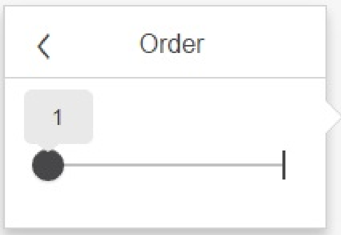
1. In Edit mode, on the toolbar, click the image icon 🖼, the media icon ▶, or the web page icon 🌐.
2. Paste the URL into the appropriate field.
3. Click and drag the object into position on the sheet.
4. Resize the object by clicking it so that handles appear ○, and drag a handle so that the object resizes as required.
5. You can layer images with other images, shapes, text, and views by positioning the image where you want it and then changing the order of the image in relation to the other objects.

   For example, you might want to have text that overlays an image.

   a) Click the text box and position it where you want it over the text.

   b) If the image hides the text, click the image, and then click ⧉.

c) To position the image behind the text, change the order slider. The higher the number, the closer the object is to the top of the screen.



Order

1

For example, if you have an image, shape, and text, and you want the image at the back and the text at the front, set the order of the image to 1, the shape to 2, and the text to 3.

Add cell values to the sheet as a cell view

You can add cell values to the sheet in a book to highlight a particular value in a view.

If the cell value changes in the cube view, the cell value changes on the sheet. If you edit the cell value on the sheet, it is updated in the cube view.

You can add comments to cell values on the sheet, and comments are inherited from the cell in the cube view, see “Comments” on page 121.

You can hold cell values by right-clicking the cell and selecting **Hold**, see “Data entry” on page 117.

You can apply conditional formatting to cells. Conditional formatting is initially inherited from the cell in the cube view, but then maintained separately, see “Conditional formats” on page 132.

You can change the cell view properties, and you can also make the cell value read-only.

Procedure

1. In Edit mode, right-click the cell that you want to add and select **Add to sheet**.

   The value is added to the sheet as a cell view that you can resize, format, and change the properties.
2. Resize the cell value by clicking it so that handles appear, then drag the handles until you get the size that you want.

   You can also position the cell view on the sheet by using the drag handle .
3. To change the format of the data in the cell, right-click the cell and select **Set format**.

   For more information, see “Change the format of data in a view” on page 58.
4. To change how the cell value properties, click the cell value, then click the **Properties** icon .

   You can change the following features:

   Exploration features

   - **Color cells by data state** - if selected, read-only cells are colored gray, and calculated cells are colored green. Clear this option to remove color.
   - **Show filters icon** - displays the filter icon in the cell view. Clicking the displayed filter icon shows the current hierarchy dimension context of the cell.
   - **Show as cell** - when checked, the cell view appears with an editable input area. When cleared, the cell view appears like the previous non-editable read only cell view.
   - **Abbreviate numbers** - abbreviates numbers. Disable to use full numbers. When you abbreviate numbers, "K" is used to represent thousands and "M" represents millions. For example, when **Abbreviate numbers** is enabled, 20,000 appears as 20K and 15,500,000 appears as 15.5M.

   General style

   The fill color, the border color, and the opacity of the cell view. If **Show as cell** is enabled, the input area is not changed based on these properties.

Synchronize

You can set the Synchronize sandbox and Synchronize dimensions options for the cell view. These settings initially reflect the settings in the source cube view. So if the Synchronize sandbox setting is cleared in the cube view, it is cleared for the cell view.

Note: You can't synchronize on the hierarchy dimensions that represent the row and column of the cell.

To find out more, see “Synchronize objects in a book or sheet” on page 116.

Text properties

The font size, font family, and font style.

5. To make the cell value read-only so that it can't be edited, click the cell value to select it, then click **Properties**, **Exploration features**, and clear **Show as cell**.

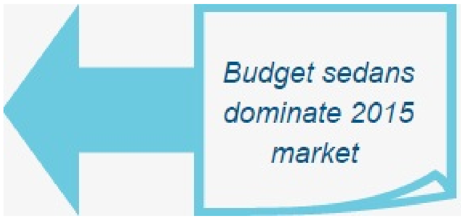
   Note: Show as cell changes the appearance of the cell view to look like the previously supported non-editable cell value.

Add shapes and text

You can add shapes and text to your Planning Analytics Workspace books. You can use these objects in combination with pictures and color to make engaging dashboards.

About this task

Suppose that you want to create a shape that looks like this:

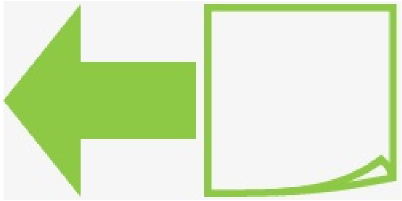


The image consists of two shapes and some text.

Procedure

1. In Edit mode, click the shapes icon on the upper right of the screen and click a shape to select it. Scroll through the shapes by clicking the arrows.

   Select this arrow shape ⬅, and this page shape □. It will look something like this:

2. Press **Ctrl** and click the shapes to select them, and then click **Properties**.
3. Select the fill color, and then click **Properties** again to close it.

   The example uses light blue, with no border.

   Tip: You can change opacity. This technique is useful if you are layering shapes over images.

4. To make an image similar to the example, click each shape and resize them using the handles ○.

5. Add text by clicking **Text** T.
6. Format text by clicking **Properties** with your text selected.

   Tip: You can layer shapes with images, other shapes, text, and views by changing the order of the shape in relation to the other objects. On the shortcut bar, click the object, then click the order icon .

Add pop-up text to books

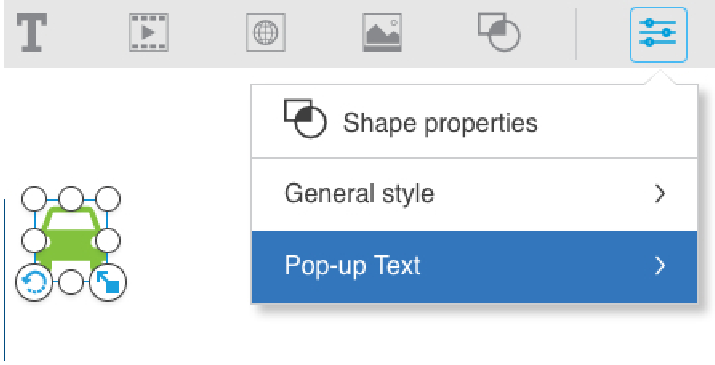
You can add explanations and guidance to books in a pop-up box. Pop-up text allows you to provide information to people about what they need to do.

Procedure

1. In edit mode, add a shape, an image, or some text, to the sheet.

   You click the shape or text to display the pop-up box.

2. Select the shape, image, or text, and click , **Pop-up Text** > **Enable Pop-up text**.



3. Select **Preferred Position**: **Top**, **Bottom**, **Left**, or **Right**. **Preferred Position** is the position of the pop-up box in relation to the shape, image, or text.

   The Preferred Position might not be possible because of the location.

   For example, if you select **Top**, but the position of the shape, image, or text that you are clicking is at the top of the screen, the pop-up box uses the opposite position.

4. Type a title and some descriptive text for the pop-up box. You can include web addresses. Web addresses are automatically formatted as URLs, so they are clickable.
5. Click **Save**, then turn off edit mode so that you can test the pop-up box.
6. Click the shape, image, or text. The pop-up box is displayed on the screen. To move the box around the screen, or resize it, first click the pin 📌. You can then move or resize the box. Click the pin again to move the box back to the original position next to the shape, image, or text.

Views

A view is a defined selection of a cube and its data that can be used for analysis, exploration, and data entry. You can also have views that are used as data sources.

Views are automatically saved to the content store as part of the book that the view was created in. The content store is where books are stored. Views that are used as data sources are stored in the TM1 database. Find out more in "Define a cube view data source" on page 250.

To quickly add a view that can be used in a book, right-click the cube in the content tree that contains the data that you want to use, and select **Add new view**, see “Add a new view to a book” on page 51.

You can also create a view from a blank template by clicking the ▦ icon in the book. See “Add a view to a book by building your view from a blank template” on page 52.

Note: When Planning Analytics Workspace opens a view, it initially fetches approximately 120 rows by 30 columns. This page size represents a compromise between performance and usability. Increasing the visible rows and columns in the view will not fetch more rows or columns from the database.

Add a new view to a book

You can create a new view of a cube and add it to a book so that you can do ad hoc analysis. A view is also called an Exploration.

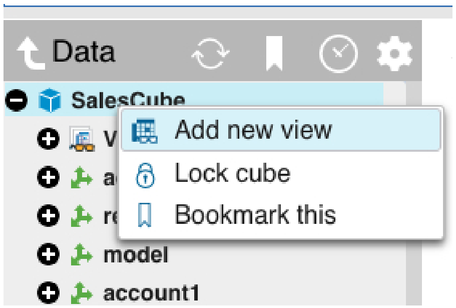
Views are automatically saved as a part of the book that they are created in. You can also save a view as a separate Planning Analytics Workspace object with a name, so it can be opened from the **Welcome** page. You can share the view with other people, and reuse the view in other books.

You can choose to save a view to the TM1 database. When a view is saved to the database, it can be used by TM1 processes as a data source from which you can extract data and create or update objects or data.

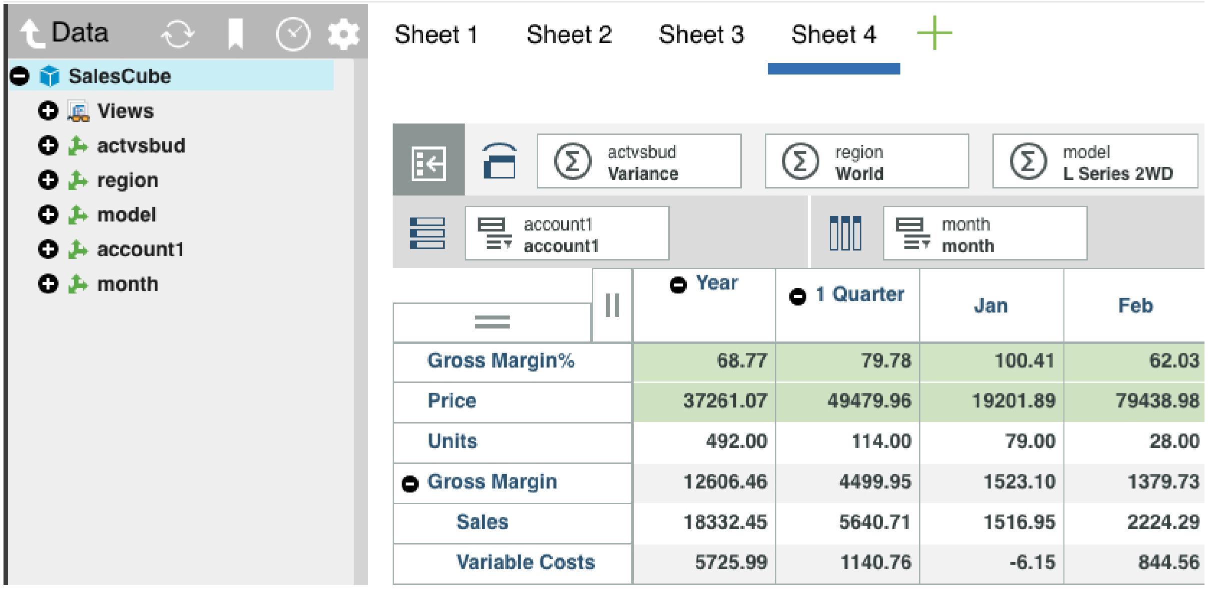
A view saved to the TM1 database is available to any Planning Analytics client that connects to the database. .

Procedure

1. Right-click the cube that contains the data that you want to view and select **Add new view**. You must be in edit mode.



   The dimensions in the cube are added to the columns, rows, and to the context area. The last dimension in the cube is added to the columns. The second to last dimension is added to the rows. All other dimensions are added to the context area.



actvsbud: Variance
region: World
model: L Series 2WD
account1: account1
month: month

	Year	1 Quarter	Jan	Feb
Gross Margin%	68.77	79.78	100.41	62.03
Price	37261.07	49479.96	19201.89	79438.98
Units	492.00	114.00	79.00	28.00
Gross Margin	12606.46	4499.95	1523.10	1379.73
Sales	18332.45	5640.71	1516.95	2224.29
Variable Costs	5725.99	1140.76	-6.15	844.56

If a private or public set that is called Default exists, the view opens with that set selected in rows, columns, and context positions. If both private and public sets that are called Default exist in a dimension, the private set takes precedence. If no sets called Default exist, all members in the dimension are used in the view in the rows and columns.

2. Save the view.

Add a view to a book by building your view from a blank template

You can create a view by positioning dimensions , hierarchies , sets, levels, or members on columns and rows on a blank template.

Before you begin

The default member insertion options determine how members are inserted into a view. To set the default member insertion options, click ⚙ in the content tree, and select one of these options:

- **Member only** (the default)
- **With children**
- **With descendants**
- **With leaves**
- **With ancestors**

Procedure

1. In a book, click ▦.
   The sheet displays a template to help you build your new view.
2. In the tree, open the cube that contains the data you want to view.
3. Expand the cube until you find the dimension, hierarchy, set, level, or member that you want on the rows. Click and hold the item and drag it to the **Drop row item here** region, the **Drop column item here** region, and **Drop context item here** region.
   For example, do the following steps:
   a. Go to the SData database, and expand the SalesCube cube.
   b. Drag the model dimension to the **Drop row item here** region.
   c. Drag the month dimension to the **Drop column item here** region.

d. In the region dimension, expand region, and in **Sets**, drag Europe to the **Drop context item here** region.

Tip: You can move the focus of the tree to the cube that you are working in by clicking this button in the view .

4. You can put more than one item on any region in your view. When you drag an extra item to a region, a vertical bar indicates the position of the item.

   Any dimensions or other items that are in a dimension that you don't add to the view are placed on the bench ... . You can drag items from the bench to use in the view later.

   For example, click the bench and drag Variance to the Context region.

5. Save the view.

Save a view
You can save a view as a Planning Analytics Workspace object or as a view on a TM1 database.

About this task

When you save a view as a separate Planning Analytics Workspace object, it can be opened from the **Welcome** page. You can share the view with other people, and reuse the view in other Planning Analytics Workspace books.

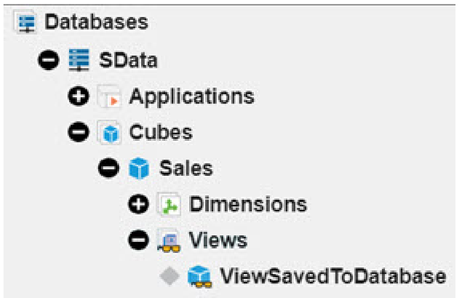
You can also choose to save a view to the TM1 database. When a view is saved to the database, it can be used by TM1 processes as a data source from which you can extract data and create or update objects or data.

A view saved to the TM1 database is available to any Planning Analytics client that connects to the database. .

Procedure

1. To save a view, click anywhere on the view to display the shortcut bar handle .
2. Click the handle, then click .
3. Choose a **Save** option.

Option	Description
Save	If the view has previously been saved as a Planning Analytics Workspace object, click **Save** to update the view definition.
Save view as	If the view has not yet been saved as a Planning Analytics Workspace object, or you want to save the view to a new Planning Analytics Workspace object click **Save view as**. a. Enter a **Name** for the view. b. Optionally, enter **Tags** and a **Description** for the view. c. Select the **Location** where you want to save the view. If you save to a location within the **Shared** folder, the view is available to other users. If you save to a location within the **Personal** folder, only you can access the view. d. Click **Save**. The saved view is available as a tile on the Planning Analytics Workspace Welcome page.
Save to Server	Click **Save to Server** to save the view to the TM1 database.

Option	Description
	a. Enter a **Name** for the view. b. Select the **Save as private view** option if you want save as a private view available only to yourself. If the **Save as private view** option is not selected, the view is saved as a public object available to any user who has at least Read access to the cube containing the view. c. Click **OK**. The view is saved to the current TM1 database. You can find the view in the content tree under the Views group for the parent cube. In this example, ViewSavedToDatabase is in the Views group for the Sales cube on the SData database.  Databases SData Applications Cubes Sales Dimensions Views ViewSavedToDatabase

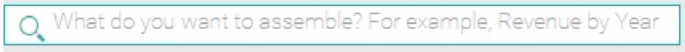
Add a view that is used as a data source to a book

You can add a view that is saved in the TM1 database to a sheet in your book, either by searching for the view in the intent bar, or by adding a view from the content tree. This kind of view is typically used by TM1 processes as a data source from which you can extract data and create or update objects or data. To find out more, see "Define a cube view data source" on page 250.

Procedure

1. To add a view to a book, you must be in Edit mode.
2. To add the view from the content tree, click the + sign next to the database.
   a. Expand **Cubes** > ***Cubename*** > **Views**.
   b. Drag the view onto the sheet.

   Tip: You can position the view on the sheet by using the drag handle in the upper left corner of the view.
3. To add the view by searching:
   a. Type `view` *`keyword`* in the intent bar.



What do you want to assemble? For example, Revenue by Year

   b. Select a view to preview. When you find the view that you want, click **Use**.

Set cube view properties

You can change the appearance of a view, change when data is refreshed, define the synchronization scope, and allow access to the Set Editor.

For example, if you want a view to have a report layout without shading and borders, you can select the report theme to apply this style.

This video shows you how you can format tables for reporting:

https://youtu.be/RCb6u5VlXR0

Procedure

1. In Edit mode, click anywhere on the cube view.
2. Click the **Properties** icon ☰.
3. Set the properties for your cube view. The properties that you can set depend on the type of cube view; explorations (table format) have different options to other kinds of chart. To find out more, see "Set visualization properties" on page 91.

   Table style

   You can configure the following features for explorations:

   - Set the **Theme** for the table.
   - You can show or hide row and column headers.

   Exploration features

   You can choose whether to display or hide the following features for explorations:

   - **Show +/-expand & collapse on rows**. Clearing this option removes the +/- indicator from row headers. This gives a more streamlined appearance, but you can't see if a member can be collapsed or expanded.
   - **Show +/-expand & collapse on columns**. Clearing this option removes the +/- indicator from column headers.
   - **Indent row levels**. Clear to remove indents from the row headers.
   - **Indent column levels**. Clear to remove indents from the column.
   - **Color cells by data state**. If selected, read-only cells are colored grey, and calculated cells are colored green. Clear this option to remove color.

   General style

   Set the **Fill color**, **Border color**, and **Opacity** for the view.

   Fill color applies to the data grid for the simple table theme. If you select dark colors such as black, red, dark blue, the text is white. With a light fill color, the text is black. If you select the grey or the blue table themes, only the area of the data grid that is not occupied by cells has the fill color applied, because these themes are not transparent.

   Chart options

   Enable **Show summary values in chart** to include consolidations in visualizations. Disable to see only leaf values in visualizations.

   Synchronize

   Set the level of synchronization you want for the view.

   Set editor

   Enable the **Allow access to Set Editor** option to allow users to access the Set Editor from the view. Disable the option to prevent access to the Set Editor. When you disable access to the Set Editor, users cannot open the Set Editor, but they can select from members of the current set.

   Text Properties

   To format text in data cells, click □.

   To format text in column headers, click ⬒.

   To format text in row headers, click ◧.

   For each of these components of the cube view, you can set font size, font family, and font style.

   When setting data cell text formats, you can apply the same settings to column and row headers by selecting the **Apply font *<characteristic>* to column and row header** option. This is the default option for font size and font family, but this option is not selected for cell font styling.

To reset the text properties to the default system font settings, click **Reset fonts**.

Views on iPads

When you open a view or websheet on an iPad, only the number of rows that fit on a single page are loaded. Depending on how the view is configured, you might see 30 - 50 rows.

As you scroll through the view, each subsequent page is loaded sequentially. This occurs even if you rapidly scroll through several pages without pausing for each page to be displayed. For example, if you are looking at a value on the first page of a view, and then scroll to page 8 of the view, pages 2 - 7 must be loaded before page 8 can be displayed. There is a slight delay as each page is loaded.

To maintain minimal load times, limit views that are accessed from iPads to 300 or fewer rows.

Change the display of data in a view

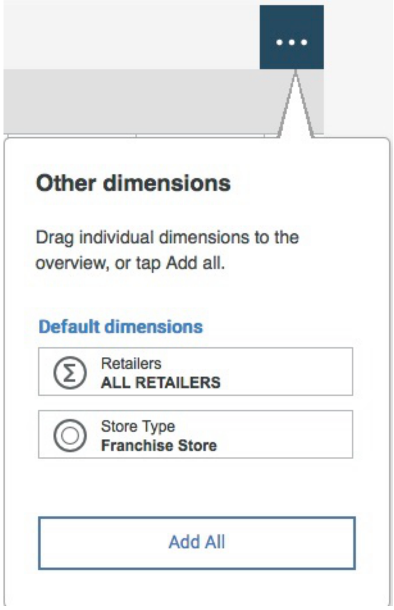
You can change the position of dimensions in a view, and you can change the presentation from an exploration (table) to a visualization (chart).

Dimensions can appear in several positions in a view: on rows, on columns, or as context. You can change the position of a dimension in a view by dragging the dimension tile and dropping it into a new position on the row, column, or context bar.

Dimensions that are used as context can be visible in the view or can be placed on the bench to save space and to simplify the appearance of the view.

Note:

If you move a dimension from a row or column to the context area, any filters, sorts, or keep, show or hide actions that have been applied to the dimension using the right click menu from the Exploration view are removed.



Other dimensions

Drag individual dimensions to the overview, or tap Add all.

Default dimensions

Retailers
ALL RETAILERS

Store Type
Franchise Store

Add All

Procedure

1. Create a book.
2. Navigate to the **GO_New_stores** database, **Base Sales Forecast** cube in the tree, and add the **All** view to the book.

   This view has one dimension on the row axis, one dimension on the column axis, and four dimensions as context.
3. Click and hold the Retailers dimension in the context area, then drop it next to the Month column dimension.

You can drag a dimension from any location in a view to a different location. When you drop one dimension directly on top of another dimension, the positions of the dimensions in the view are switched. When you drop one dimension next to another dimension, both dimensions appear in the same location in the view.

4. Click anywhere on the view, and then click .

   The shortcut bar appears. The actions available on the shortcut bar vary depending on the mode you are in.

5. Click to swap the position of the row and column dimensions.

   Tip: You can also type **sw** in the snap command box.

6. Click in a cell, click , then select a visualization.

Hide row or column headers in a view

When a view is present as an Exploration (table), you can hide row or column headers in the view. This is useful if you want to display an Exploration while minimizing its size within a sheet.

About this task

Hiding row and column headers applies only to Explorations. For other visualization types, you can set visualization properties to enable or disable the display of axis labels.

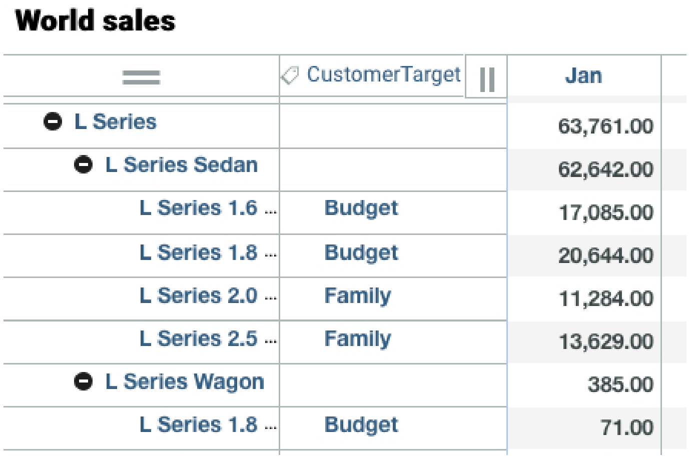
Procedure

1. In Edit mode, click the Exploration.
2. Click the **Properties** icon , then **Table style**.
3. Select one or both of the **Hide** options to hide headers in the view.
4. Clear one or both of the **Hide** options to display headers in the view.

Show member attributes in the cube view

Some dimension members have attributes. Attributes help to explain or describe a member in a dimension. You can show member attributes in a table.

For example, suppose you have a dimension with car models with an attribute called CustomerTarget, and you want to see the customer target for each car model in the table. You can choose to show or hide this attribute.



World sales

	CustomerTarget	Jan
⊖ L Series		63,761.00
⊖ L Series Sedan		62,642.00
L Series 1.6 ...	Budget	17,085.00
L Series 1.8 ...	Budget	20,644.00
L Series 2.0 ...	Family	11,284.00
L Series 2.5 ...	Family	13,629.00
⊖ L Series Wagon		385.00
L Series 1.8 ...	Budget	71.00

Procedure

1. Right-click either the row selector =, or the column selector ||, and select **Show attributes**.
2. Select the attribute that you want to show in the **Available attribute** pane, and click → to move the attribute into the **Selected attribute** pane.

   You can select several attributes and you can choose the order in which the attributes are displayed by moving them using the up and down arrows.
3. Click **OK** to save your choices and to return to the table.
4. You can sort the attributes in either ascending or descending order. Right-click either the row selector, or the column selector, and select either **Sort ascending** or **Sort descending**.
5. To hide attributes, right-click the row or column selector, and select **Hide attribute**.

Change the format of data in a view

You can set the format of data in a view, overriding existing formatting in a dimension. A number of built-in formats are available, or you can specify a custom format.

Note: Data in a view is displayed in accordance with your browser locale setting, using the decimal and thousands separators for your locale. You can double-click a cell to view the raw data value as it exists on the Planning Analytics database, which uses a period as the decimal separator and omits a thousands separator. The raw value may include more decimal places than are displayed in the cell. When you enter data directly in a cell, you must use the decimal separator for your browser locale.

When you set the format in the view, the raw data format or cell data type is not changed. You can reset the format in the view to the underlying raw data format by selecting **Use database format**.

Formats that are saved to a view in IBM Planning Analytics Workspace are not available if the view is opened in IBM Planning Analytics for Microsoft Excel.

Generally, when more than one format is set in a view, the format that is applied to a cell is determined in the following order of precedence:

1. The column member inner to outer axis (column dimension in the overview, from right to left).
2. The row member inner to outer axis (row dimension in the overview, from right to left).
3. Formatting applied to the entire view from the **Format** button on the shortcut bar.
4. Context dimensions in the overview, from right to left.
5. Bench dimensions from bottom to top.

There are some unique format interactions in the case of column or row formatting versus entire view formatting. When you apply view formatting, it overrides all column or row formatting. However, any column or row formatting that is applied **after** the view formatting is applied will take precedence, in accordance with the order of precedence described above.

You can have two copies of the same view and apply different formatting to each view. For example, you have a view of a bank in Canada that shows Total Current Assets in thousands. You can make a duplicate of this view, and change it to show worldwide bank data. When you duplicated the view, the formatting was the same in the copy. But you can change the formatting in the copy to show the Total Current Assets in millions.

The built-in formats are:

General

Displays numbers without commas to separate digits to the left of the decimal point. Negative values are prefixed with a minus sign (-).

`-1234.57`

Fixed
Displays numbers without commas to separate digits to the left of the decimal point. Negative values are surrounded by parentheses.

`(1234.57)`

Comma
Commas separate every third digit to the left of the decimal point.

`(1,234.57)`

Rounded
Commas separate every third digit to the left of the decimal point, rounded up to the nearest whole number. You can see the actual value by right-clicking in the cell.

`(12,346)`

Percentage
Multiplies numbers by 100 and displays a following percent sign (%). Digits to the left of the decimal point do not use commas, and negative values are prefixed with a minus sign (-).

`-123456.70%`

Scientific
Displays numbers in scientific notation. Scientific notation is a way of expressing large or small numbers. For example, the number 123,000,000,000 can be written as 1.23E+11.

Negative values are prefixed with a minus sign (-).

`-1.2E+3`

Accounting
Displays numbers with currency symbols and decimal points in a column. Negative values are surrounded by parentheses.

`$(1,234.57)`

Currency
Displays numbers with the currency symbol that is specified for your computer. Commas separate every third digit to the left of the decimal point. Negative values are surrounded by parentheses.

`($1,234.57)`

Currency (rounded)
Displays numbers with the currency symbol that is specified for your computer. Commas separate every third digit to the left of the decimal point, rounded up to the nearest whole number. You can see the actual value by right-clicking in the cell. Negative values are surrounded by parentheses.

`($1,235)`

Date
Gives you a date picker, and displays dates in a predefined format: `mm/dd/yyyy`.

`01/23/1989`

Time
Displays time in a predefined format: `hh:mm:ss`.

`12:30:00`

Thousands
Displays thousands as K.

`12K`

Millions
Displays millions as M.

`12M`

Procedure

1. To set format on a row or column:
   a) Right-click the column or row header, and click **Set format**.
   b) Select the format, such as **Currency**.

      You can apply formatting to both columns and rows. Column formatting takes precedence over row formatting.

   c) If none of the built-in formats are suitable, select **Custom format** and type the pattern for the custom format.

      Learn more in "Custom formats in views" on page 60.

   d) To reset the format that is used to the database default, select **Use database format**.
2. To set format on the entire view:
   a) Click anywhere on the view to display the shortcut bar handle ▼.
   b) Click the handle, then click ☰.
   c) Select the format, such as **Currency**.
   d) If none of the built-in formats are suitable, select **Custom format** and type the pattern for the custom format.

      Learn more in "Custom formats in views" on page 60.

   e) To reset the format that is used to the database default, select **Use database format**.

Custom formats in views

You can change the way that data is formatted in a view by creating a custom format, if the data format that you want isn't available in the built-in formats.

Creating a custom format can be useful when working with views. For example, you might want to specify a particular currency symbol, such as the Euro (€), or use a different date or time format.

Important: When you set the format in the view, the underlying TM1 format or TM1 cell data type is not changed. You can reset the format in the view to the underlying TM1 format by selecting **Use database format**.

Custom formatting in IBM Planning Analytics Workspace views uses ICU syntax. For detailed information, go to the following website: http://icu-project.org/apiref/icu4c/classDecimalFormat.html, and search for *Special Pattern Characters*.

Custom formats use a specific pattern, and many characters in a pattern are taken literally, so they appear unchanged. The following example shows how you can insert the word Total into rows or columns, and it also shows how to insert a currency symbol (€) into the value.

Example custom format	Result
Total: €#,##0;(€#,###)	Value: `28.50` Displays: `Total: €29`

Notice that the # and 0 characters are used to represent digits. The # and 0 characters are examples of special pattern characters. The following table provides a summary of the special characters that are used in custom formats.

Table 2. Special pattern characters

Symbol	Meaning
0	**Digit** Can be used in combination with #. For example, #,##0.00;(#,##0.00) Displays insignificant zeros if a number has fewer digits than there are zeros in the format string. If a number has more digits to the right of the decimal point than there are placeholders in the format string, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. **Examples:** Custom format: 0.00 Value: 23.896 Displays: 23.90
#	**Digit, zero shows as absent** Can be used in combination with 0. # displays only significant digits. In the value .90, the 0 is considered insignificant. The value is displayed as .9 when # placeholder is used. If a number has more digits to the right of the decimal point than there are placeholders in the format string, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. **Example:** Custom format: #.## Value: 123.896 Displays: 123.9
1-9	**'1' through '9' indicate rounding**
@	**Significant digit** **Example:** Custom format: @@@ Value: 123.896 Displays: 124
.	**Decimal separator or monetary decimal separator**
-	**Minus sign**
,	**Grouping separator** **Example:** Custom format: #,##,##0 Value: 123456789 Displays:12,34,56,789

Table 2. Special pattern characters (continued)

Symbol	Meaning
E	**Separates the mantissa and exponent in scientific notation**
+	**Prefix positive exponents with the localized plus sign**
;	**Separates positive and negative subpatterns** If you don't specify how negative numbers are shown, positive formatting is used, prefixed by the minus sign. **Example:** Custom format: #,##0.00;(#,##0.00) Positive numbers: `123,456,789.00` Negative numbers: `(123,456,789.00)`
%	**Multiply by 100 and show as percentage**
'	**Used to quotation mark special characters in a prefix or suffix** **Example:** Custom format: `"'#'#"` Value: 123 Displays: `"#123"`. To create a single quotation mark itself, use two in a row: `"# o''clock"`.

Date and time examples

For custom date formats, you use a date pattern. In a date pattern, strings of characters are replaced with date and time data.

Example	Result
hh:mm a	• 12:00 AM • 06:00 PM
EEE, MMM d, ''yy	Mon, Oct 30, '15

For more information, go the following website: http://userguide.icu-project.org/formatparse/datetime, and look for *Date/Time Format Syntax*.

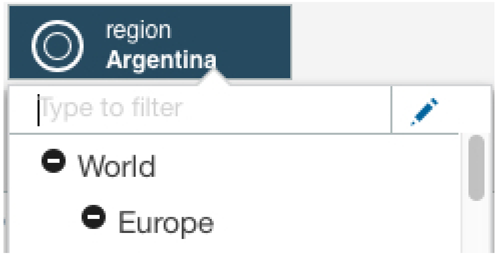
Change the members in a view

You can change the members that are visible in a view. The context area of a view includes a single member from each dimension in the view. To change your view of data, choose a new member for a context dimension.

The rows and columns in a view display one or more members from a dimension in a set. A set is a limited number of members in a dimension. You can change the members in a set to see a different view of your data.

Procedure

1. To change the member of a context dimension, click the dimension tile, then click the member that you want to use in the view.



You can also filter for members that match a string you type in the **Filter** field, which is active when you initially click a tile. As you type, members that match your string appear, and you can select the one you want. You can filter in tiles on the context area, on the bench, or on the row/column axes.

2. To change the member of a dimension that is on the bench, click ⋯, click the dimension name, then click the member that you want to use in the view.
3. To view a single member from the current set, double-click the member name on the row or column.

   The view displays only the member that you click.
4. To modify the set to display multiple members, click the dimension tile.

What to do next

You can modify a set, or create a new set of members to limit the number of members that you can see or select in a view. This is very useful if you have a large dimension. To find out more, see "Create and edit sets" on page 141.

Visualizations in Planning Analytics Workspace

You can use any of the visualizations that are described here to present data in Planning Analytics Workspace.

To change the visualization type for any view in a book:

1. Click the view.
2. Click the **Change visualization** icon. The label next to the icon indicates the type of visualization currently displayed for the view.
3. Click **All visualizations**.
4. Select the visualization you want to apply to the view.

Note: In some cases when you attempt to convert an Exploration to a visualization, you might receive an error indicating that '

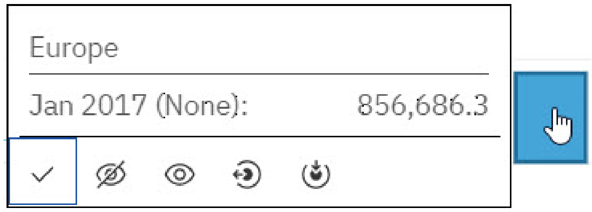
```
Data for <member> is missing.
```

' When this happens, Planning Analytics Workspace cannot construct the requested visualization.

Here are some use reasons why this error can occur:

- Zero suppression is turned on in the view, resulting in the removal of a column member that is required to create the visualization
- A calculation was removed, resulting in the removal of a column member that is required to create the visualization
- Any other view action that could cause column members required by the visualization to be unavailable
- Insufficient TM1 permissions in the view. For example, if User1 creates a view that includes members for which User2 does not have at least Read permission, User2 may encounter an error when trying to create a visualization.

When you right-click on a visualization member, you'll see the full abbreviated value of the member, along with several options to interact with the visualization.



Europe

Jan 2017 (None): 856,686.3

Hide
: Hide the member in the visualization.

Unhide all
: Reveal all hidden members in the visualization.

Drill up
: Collapse the member to show its immediate parent.

Drill down
: Reveal the members of a consolidation. You cannot drill down on a leaf member.

Create a visualization directly in a book

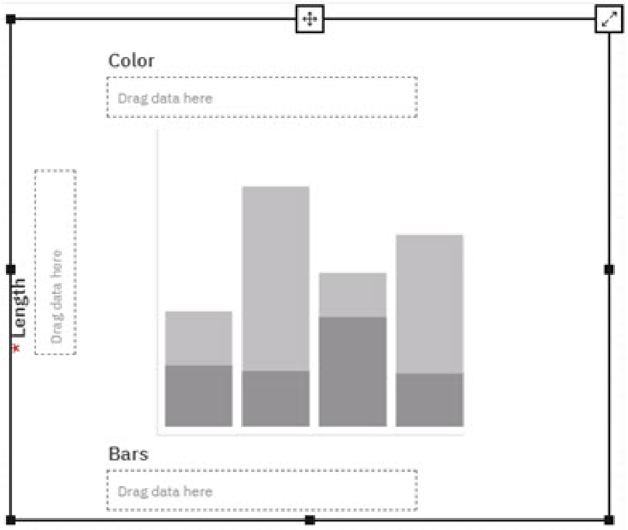
You can build a visualization from scratch in a book, without having to convert an existing Exploration.

About this task

You must be logged in as an analyst, modeler, or administrator to create a visualization.

Procedure

1. Create a new book or open the book where you want to create the visualization.
2. If necessary, click to enable Edit mode.
3. Click the **Visualizations** button, then click the type of visualization you want to create.
   A template for your visualization is inserted into your book.

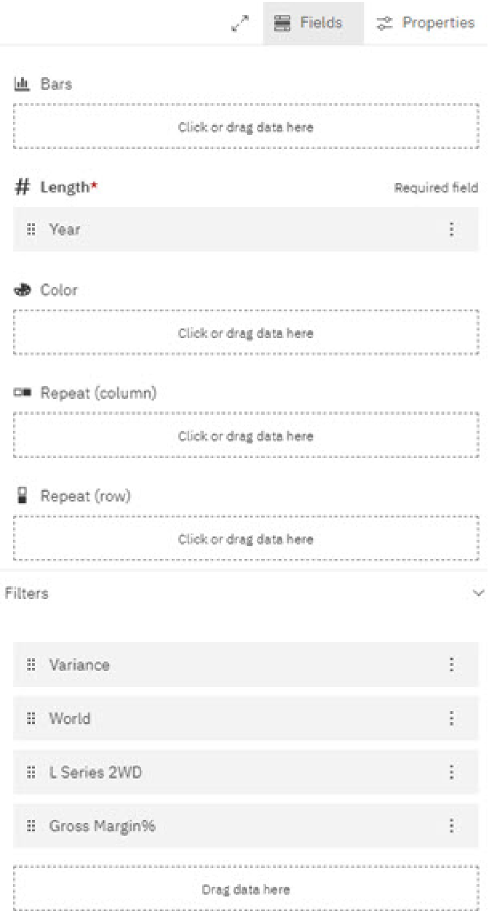
   The template shows an example of the type of visualization you are building. There are several **Drag data here** drop zones representing fields in your visualization. Required fields are identified by an asterisk.

   For illustration purposes, we'll build a **Stacked column** visualization, but the principles described here apply to all visualizations.
4. Click the **Database** icon to open the Data tree.

5. Locate the cube that you want to use as the data source for the visualization.
6. Expand the cube and identify the dimension that you want to insert in one of the required fields of the visualization. Alternatively, you can identify a subset or single member to insert into the visualization.
7. Drag and drop the dimension (or subset or member) onto the **Drag data here** drop zone for a required field.

   At this point, you've established a relationship between the cube and the visualization. You can now use the other dimensions of the cube to complete the visualization.

   Take a look at the **Fields** tab and you'll notice that the dimension that you dropped onto the visualization is in the appropriate field. All other dimensions from the cube are positioned as Filters. Filters set the context for a visualization, just as dimensions on the bench determine the context for an Exploration.



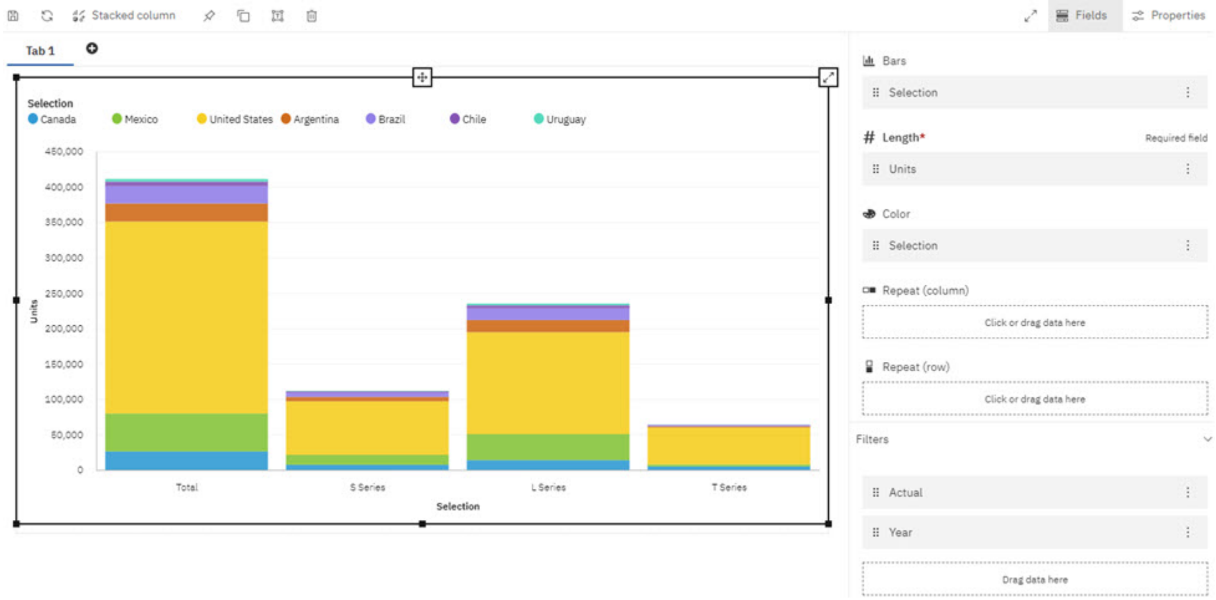
   You can finish building your visualization directly in the Fields tab.
8. Drag and drop a dimension from the **Filters** list onto one of the fields for the visualization.

   You'll immediately see the impact of your action in the visualization. Continue moving dimensions from the Filters list to the visualization fields until your visualization is configured the way you want it. At a minimum, you have to define the required fields for the visualization, but the more fields you define the greater the detail you can see in the visualization.

   You can ignore the **Repeat (row)** and **Repeat (column)** fields for now, we'll discuss those later.

   You can modify the visualization configuration by dragging and dropping dimensions from one field to another. When you drop a dimension on an occupied field, the dimension positions are switched. For example, if you drag *DimensionA* from the Color field onto *DimensionB* on the Bars field, you end up with *DimensionA* on the Bars field and *DimensionB* on the Color field.

   Here's an example of a fully configured Stacked column visualization. You can see that all fields other than **Repeat (row)** and **Repeat (column)** are defined.



Bars
Selection

Length* Required field
Units

Color
Selection

Repeat (column)
Click or drag data here

Repeat (row)
Click or drag data here

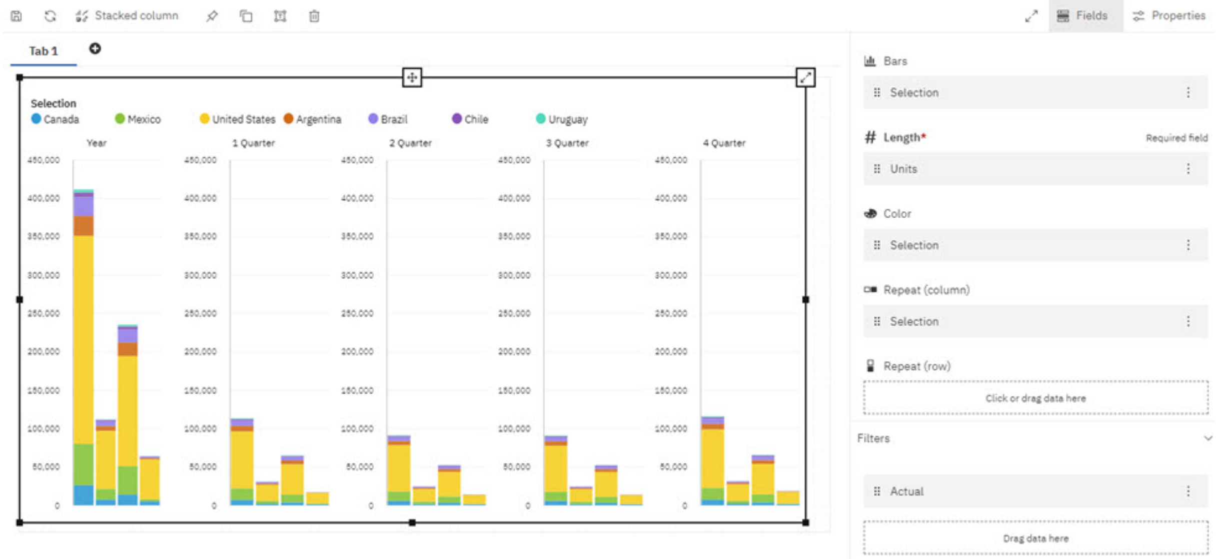
Filters
Actual
Year
Drag data here

9. Click on any occupied field to further refine your visualization.

   When you click on an occupied field, including fields in the Filters list, you can select a different member from the current set or a defined dimension level. You can also click **Edit this set** to open the Set Editor and modify the set as desired.

10. Now that you've got your visualization dialed in, you can use the **Repeat (row)** or **Repeat (column)** fields to repeat the visualization for multiple members. When you drag a dimension to the **Repeat (row)** or **Repeat (column)** field, the visualization is repeated for each member in the dimension set, oriented along either rows or columns.

    For example, the Stacked column visualization above shows information for the entire year, but you might want to repeat the visualization for each quarter in the year. In this case, you'd drag Year from the Filters list to the **Repeat (column)** field. You might have to modify the set slightly, but the result is a repetition of the visualization for each quarter.



Bars
Selection

Length* Required field
Units

Color
Selection

Repeat (column)
Selection

Repeat (row)
Click or drag data here

Filters
Actual
Drag data here

Visualizations available in Planning Analytics Workspace

The following visualizations can be created in Planning Analytics Workspace.

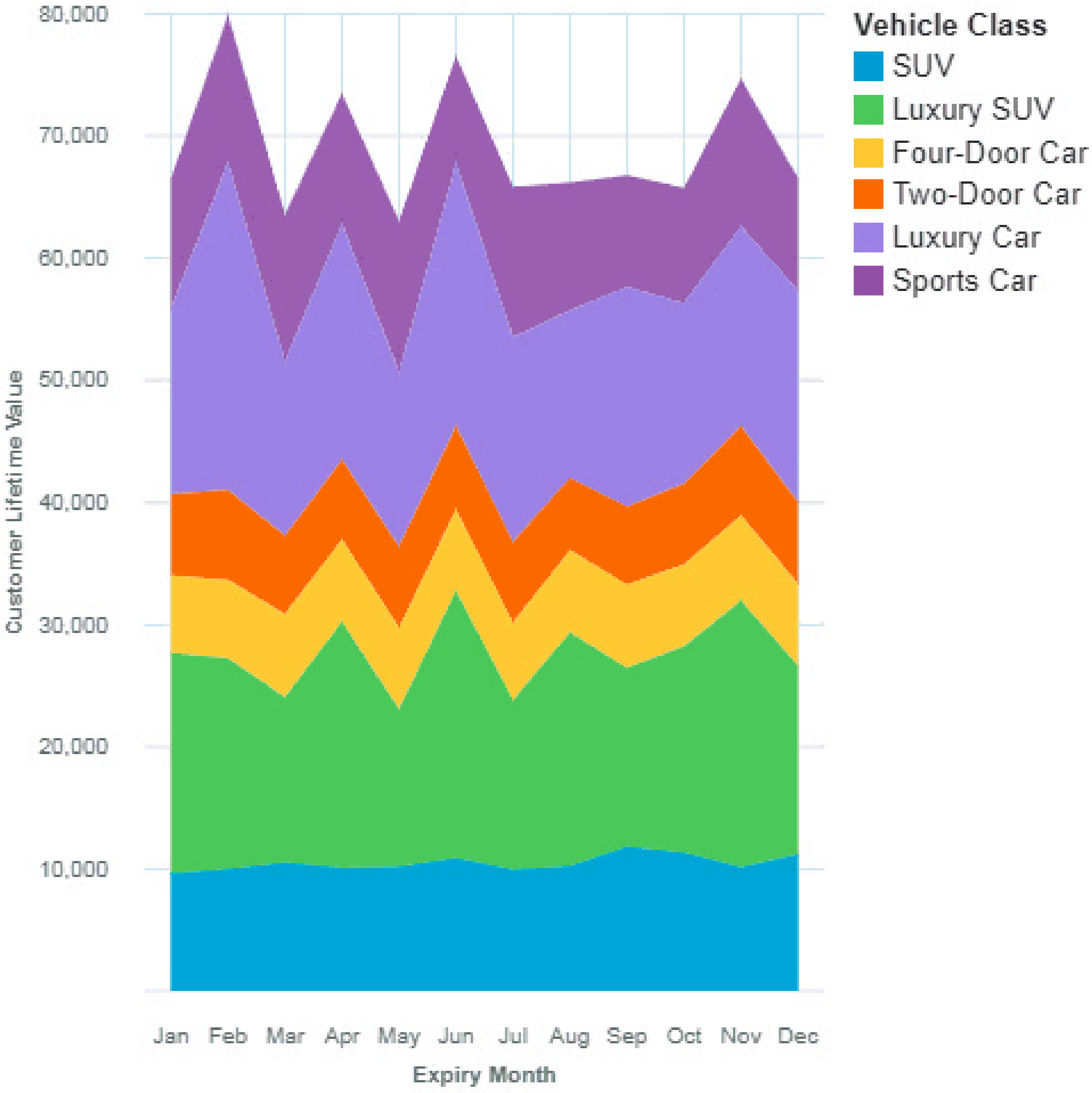
Area
Use an area visualization to emphasize the magnitude of change over time.

Area charts are like line charts, but the areas below the lines are filled with colors or patterns. Stacked charts are useful for comparing proportional contributions in a category. They plot the relative value that each data series contributes to the total.

Because an area visualization stacks the results for each column or item, the total of all results is easily seen.

For example, an area visualization is excellent for looking at revenue over time across several products.

This area visualization shows the customer lifetime value for each vehicle class per month. Because the area visualization stacks the results, you see the totals for each month.

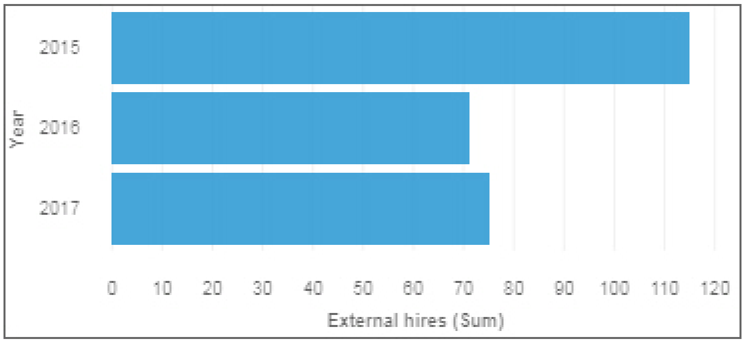


Bar

Use a bar visualization to compare values by one or more columns, such as sales for products or sales for products each month.

Bar visualizations use horizontal data markers that are arranged in groups to compare individual values. You can use bar visualizations to compare discrete data or to show trends over time.

A bar visualization can show change over a specific time period or can compare and contrast two or more columns in a time period or over time. If there are so many bars that the labels are impossible to read, filter the data to focus on a subset of the data or use a tree map.



Use the **Target** field to show measures that need to be compared against a target value.

Use the **y-start** field to define where the measure must start.

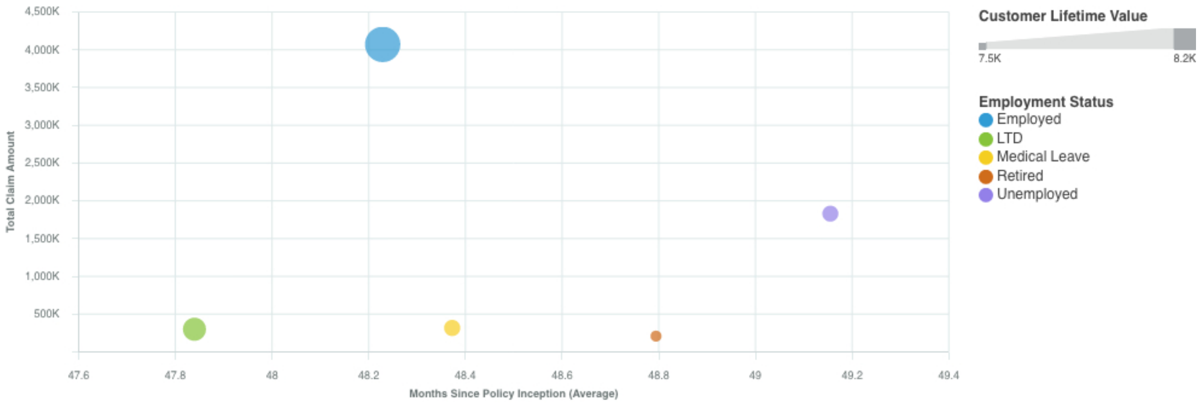
Bubble

Use a bubble visualization to show relationships among columns that contain numeric values, such as revenue and profit.

A bubble visualization uses data points and bubbles to plot measures anywhere along a scale. One measure is plotted along each axis. The size of the bubble represents a third measure. Use bubble visualizations to represent financial data or any data where measure values are related.

The bubbles are in different sizes and colors. The x-axis represents one measure. The y-axis represents another measure, and the size of the bubbles represents the third measure. In the example shown below, color is represented by an identifier.

The example that is shown represents the months since the policy inception.



Bullet

Use bullet charts to show measures that need to be compared against a target value.

In a call center, a bullet chart can be used to measure metrics like call volume, call answer speed, and percentage of abandoned calls.

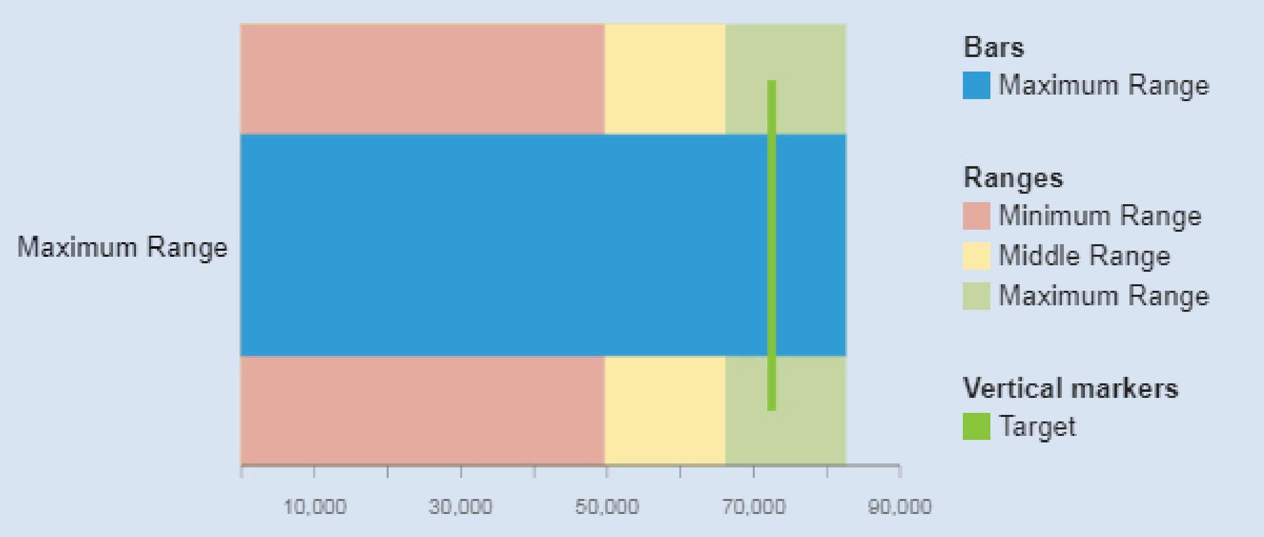
In manufacturing, a bullet chart can be used to track metrics like number of defects and orders that are shipped.

In a fitness context, a bullet chart can be used to measure metrics like steps that are taken and calories that are burnt.

Bullet visualizations compare an actual measure (the bullet) to targeted measure (the target). Bullet visualizations also relate the compared measures against colored regions in the background that provide more qualitative measurements, such as good, satisfactory, and poor. Bullet visualizations can be shown at small sizes while still effectively conveying information.

A bullet visualization features a single, primary measure. For example, current year-to-date revenue. And compares that measure to one or more other measures to enrich its meaning. For example, compared to a target. The primary measure is displayed in the context of a qualitative range of performance, such as poor, satisfactory, and good.

Make sure the minimum, medium, and maximum ranges relate to the actual and target measure.

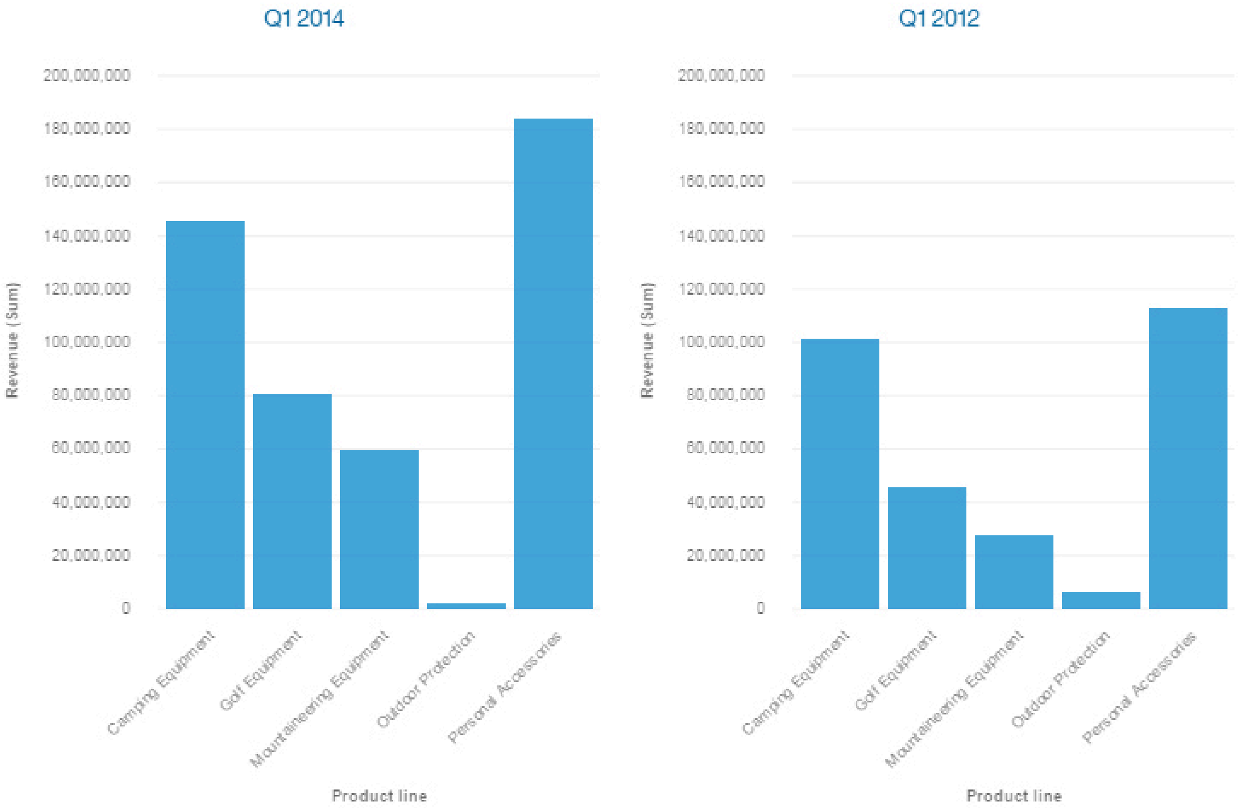


Column
Use a column visualization to compare values by one or more columns, such as sales for products or sales for products each month.

Column visualizations use vertical data markers that are arranged in groups to compare individual values. Use column visualizations to compare discrete data or show trends over time.

A column visualization shows change over a specific time period or can compare and contrast two or more columns in a time period or over time. If there are so many bars that the labels are impossible to read, filter the data to focus on a subset of the data or use a tree map.

For example, revenue for each product line is grouped by quarter, which emphasizes performance in each quarter.



Use the **Target** field to show measures that need to be compared against a target value.

Use the **y-start** field to define where the measure must start.

Data player

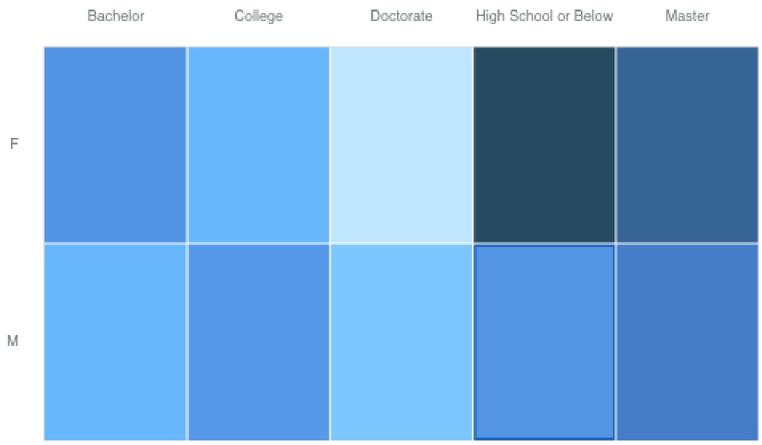
Use a data player to see an animation of the impact of a column on the other visualizations.

Heat map

Use a heat map visualization to visualize the relationship between columns, represented in a matrix type view.

A heat map visualization uses color and intensity of the color to show the relationship between two columns.

For example, this heat map visualization shows the average customer lifetime value by gender and education.



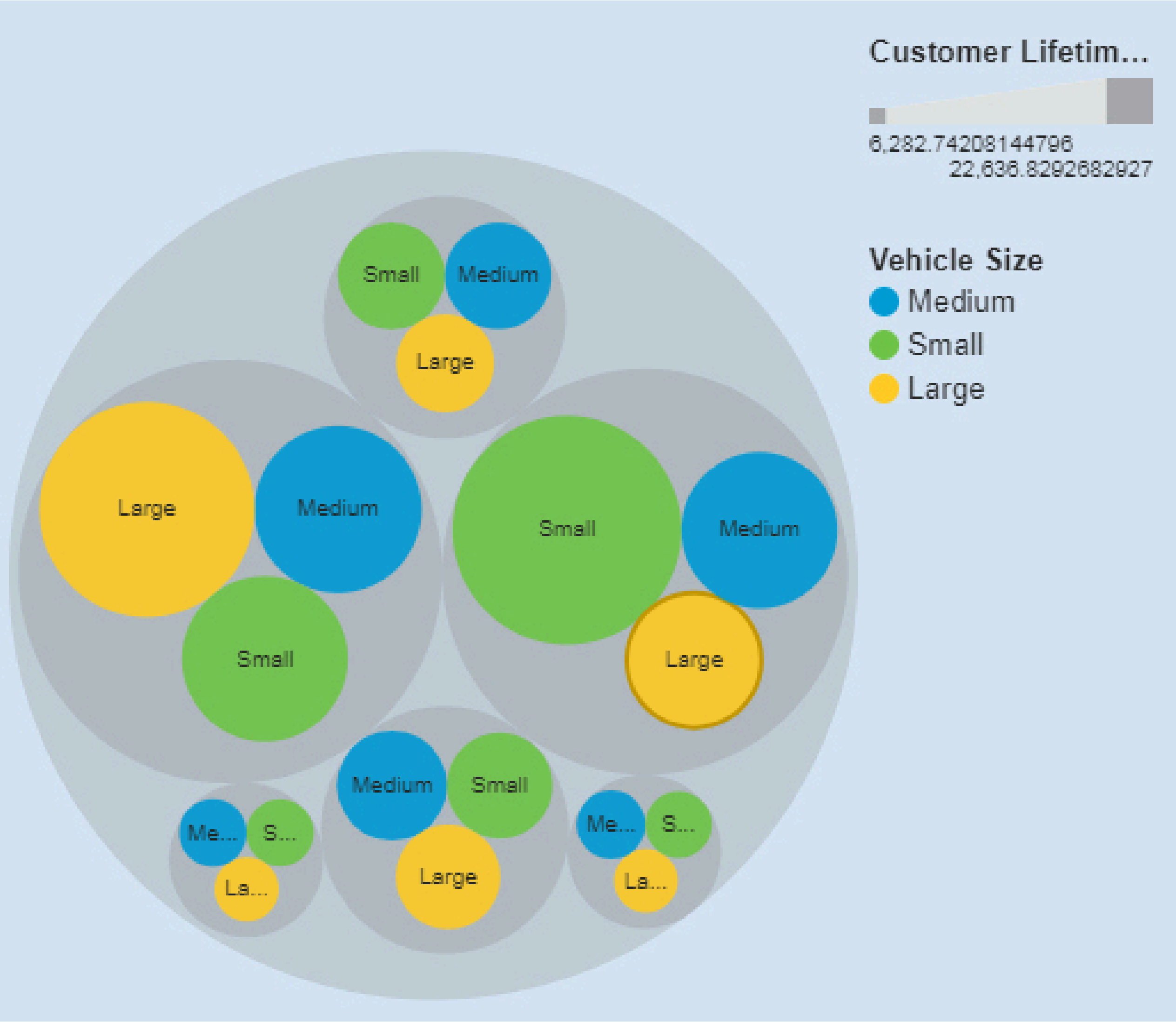
Hierarchy bubble

Use a hierarchy bubble visualization when you want to show relationships among columns that contain values, such as net loss. It is similar to the bubble visualization but the bubbles are tightly packed instead of spread over a grid. The bubbles use nesting to represent the hierarchy. A hierarchy bubble visualization shows a large amount of data in a small space.

The size of each bubble shows a quantitative dimension of each data point. It shows many levels within a hierarchy and relationships between groups based on assigned attributes. It uses bubble size and color to convey comparative information about categories.

The bubbles are in different sizes and colors.

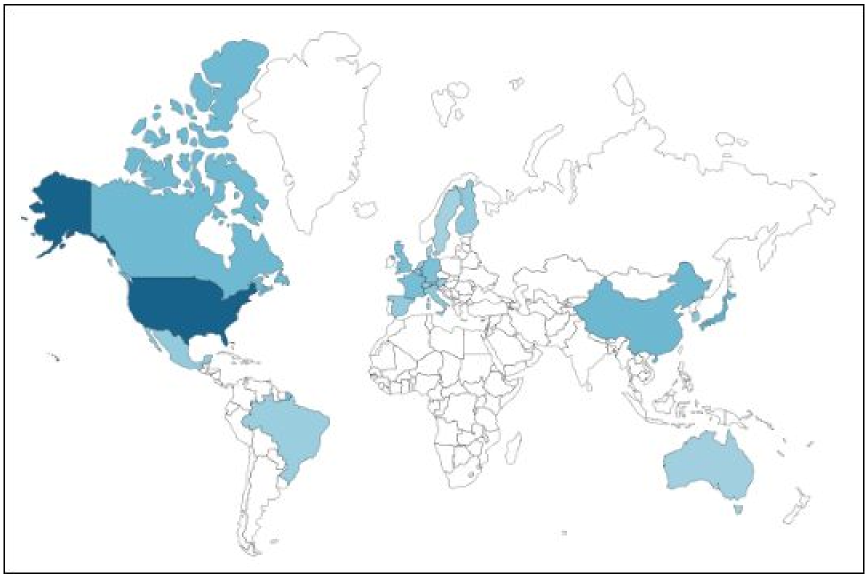
For example, this hierarchy bubble visualization shows customer lifetime value by vehicle class per vehicle size. Each bubble is a different vehicle class in one of the three vehicle size. The size of each bubble is determined by the customer lifetime value of that vehicle class. The colors of the bubbles are determined by the vehicle size.



Legacy map

Use a legacy map when you want to see patterns in your data by geography.

For example, this legacy map visualization shows revenue by retailer country with the darker color indicating higher revenue.



To create a map visualization, your view must contain a defined geography dimension.

The geography dimension must contain elements that are recognizable as geographic entities. To determine if a dimension is mappable, Planning Analytics Workspace analyzes a sample of members in the defined geography dimension, looking for recognizable place names. If 80% or more of the members in the geography dimension are recognized as geographic entities, a map is generated.

For more information on the languages and geographic entities supported in map visualizations, see "Map reference info" on page 647.

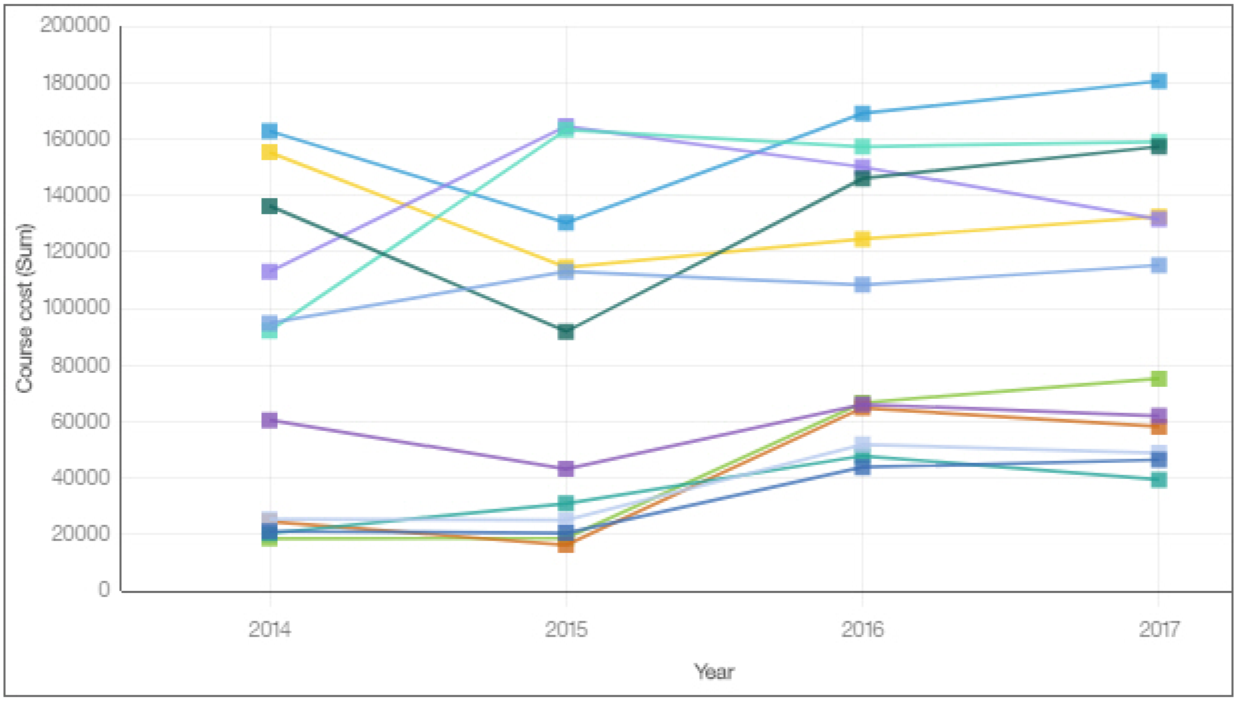
Line

Use a line visualization to show trends over time.

A line visualization can compare trends and cycles, infer relationships between variables, or show how a single variable is performing over time.

For an effective line visualization, use a time column in the x-axis, such as years, quarters, months, or days. If the x-axis shows something else, such as Canada, Netherlands, UK, and US, use a bar or column visualization.

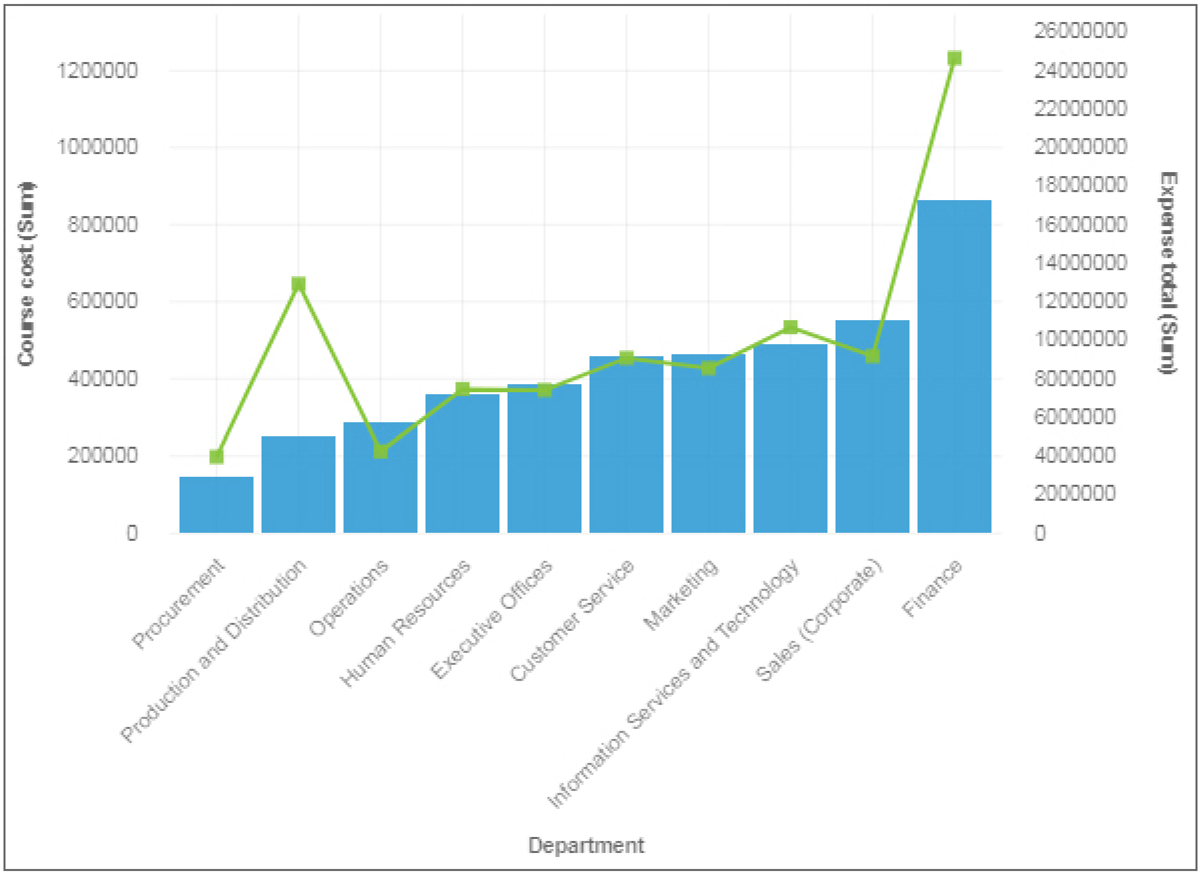
For example, this line visualization shows the trend in course costs by department over year.



Line and column

Use a line and column visualization to highlight relationships between multiple data series by combining bars and lines with one visualization.

For example, this line and column visualization shows the relationship between course cost and expense totals by department.



List

Use a list visualization to create an overview of data in an hierarchical way.

Marimekko

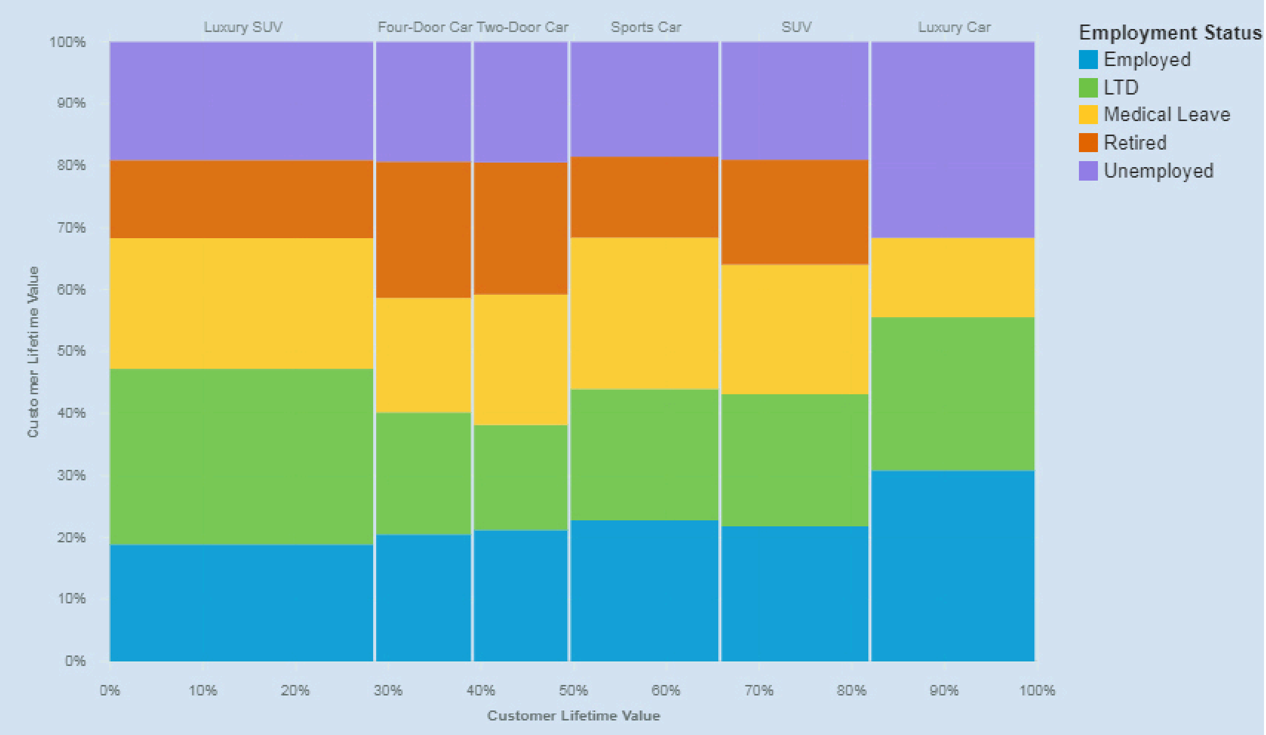
A marimekko visualization is similar to a stacked column visualization. It shows data through varying heights and includes an added dimension of data through varying column widths. The width of the columns is based on the value that is assigned to the width field. Individual segment height is a percentage of the respective column total value.

You can quickly spot large segments, such as a specific vertical that has a large share of a region. You can also identify white space such as an under-represented vertical in a specific region.

The marimekko visualization is useful for part-to-whole comparisons, where you need to show an extra measure/variable.

The marimekko visualization allows data to be depicted along two dimensions simultaneously. For example, market segments are often arrayed along the x-axis, with the width of each column corresponding to the financial value of a segment. You use marimekko visualizations in cases, for example, where you want to show the revenue contribution per product line. Or the gross domestic product per country.

The following example shows the contribution of customer lifetime value and employment status in different vehicle classes.



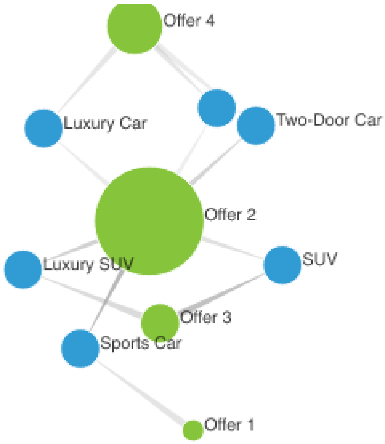
Network

Use a network visualization when you want to see the connections among columns in your data A network visualization is a good choice to show connections, networks, and points of intersection.

Network visualizations display a set of nodes, represented by symbols, and links, represented by paths, to show the relationship between entities or items.

Use the **From** and **To** fields to define the relationship that you want to investigate.

For example, a network visualization can show offer acceptance by Vehicle Class.

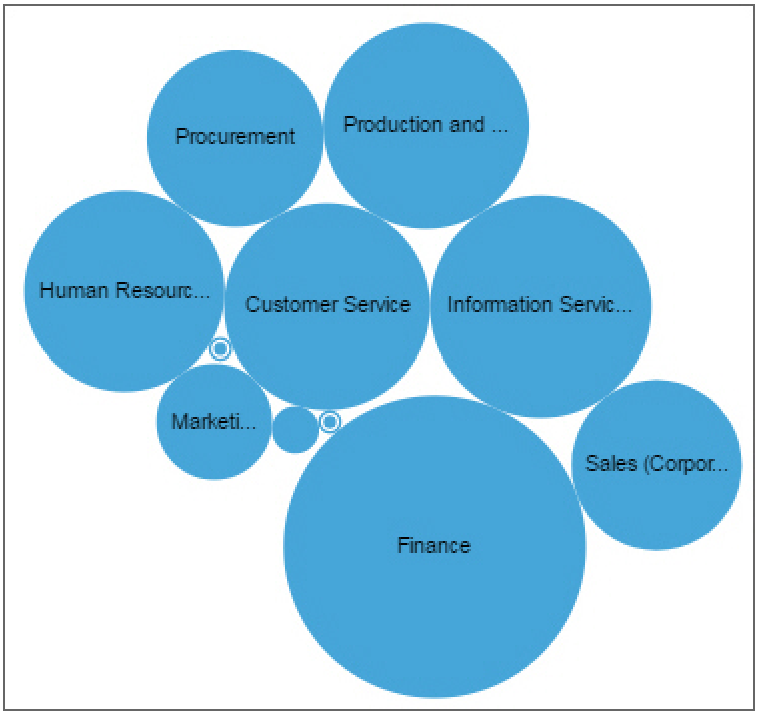


Packed bubble

Use a packed bubble visualization when you want to show relationships among columns that contain numeric values, such as revenue. It is similar to the bubble visualization but the bubbles are tightly packed instead of spread over a grid. A packed bubble visualization shows a large amount of data in a small space.

The bubbles are in different sizes and colors.

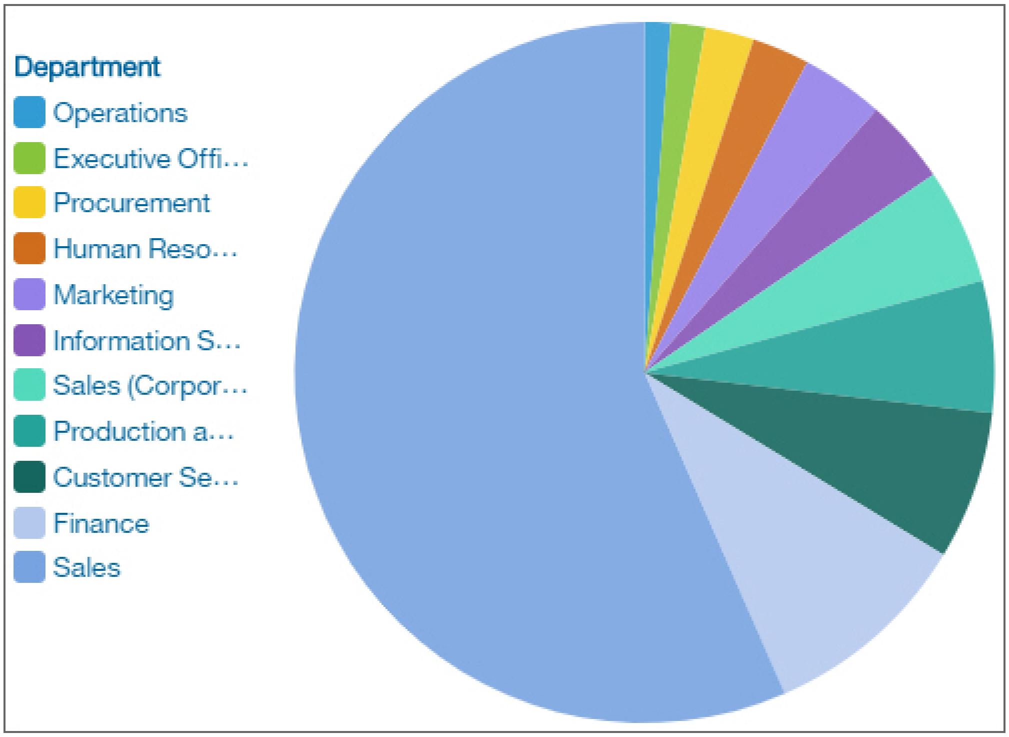
For example, this packed bubble visualization shows external hires by department. Each bubble is a different department. The size of each bubble is determined by the number of external hires for that department.



Pie

Use a pie visualization to highlight proportions. Each slice shows the relative relationship of each part to the whole.

For example, this pie visualization shows the number of course days for each department.



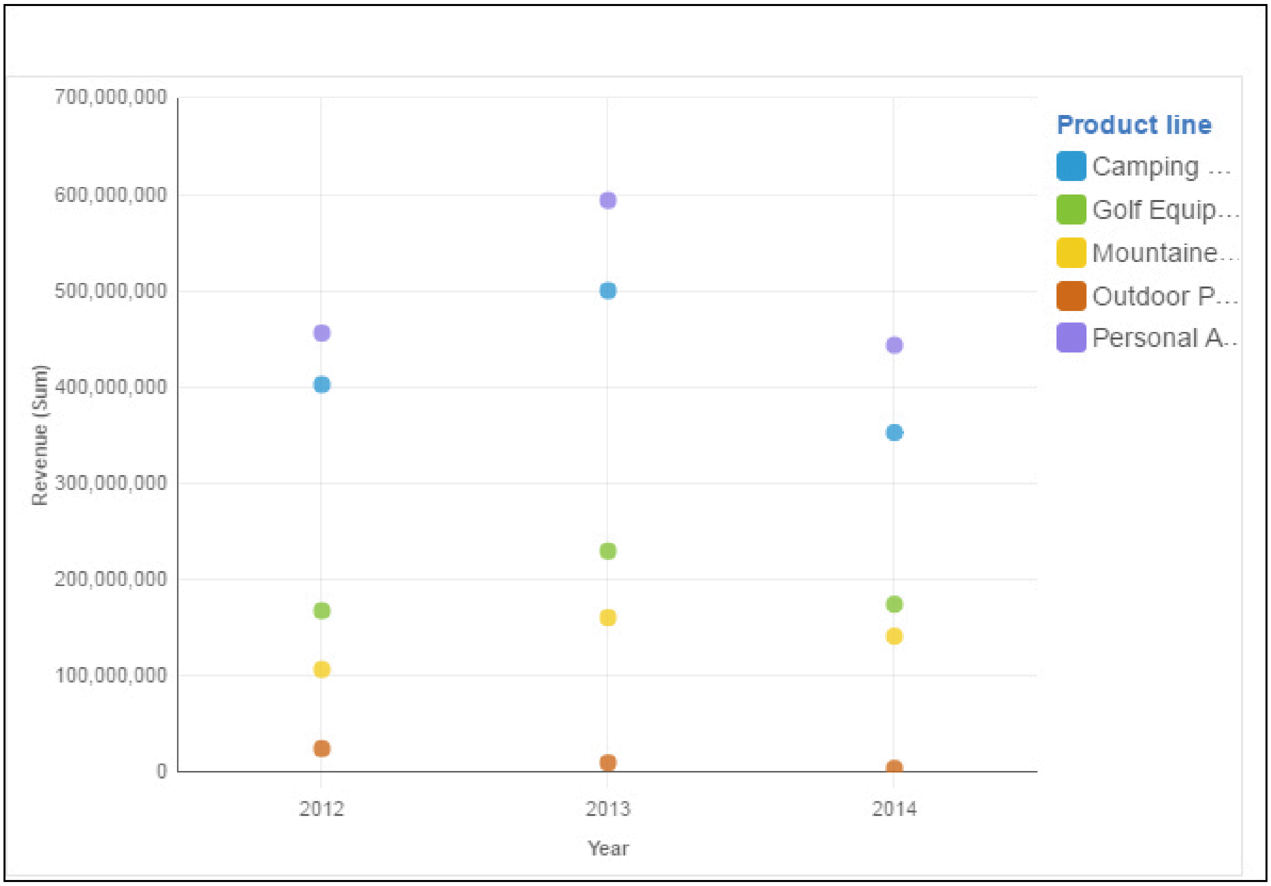
Point
Use a point visualization to show trends over time.

A point visualization can compare trends and cycles, infer relationships between variables, or show how a single variable is performing over time.

A point visualization is like a line chart without the connecting lines.

For an effective line visualization, the x-axis should show time, such as years, quarters, months, or days. If the x-axis shows something else, such as Canada, Netherlands, UK, and US, use a bar visualization.

Data values are plotted vertically.

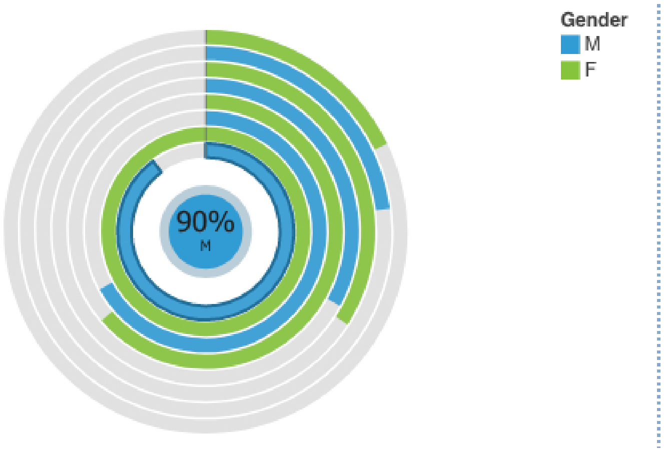


Radial

In a radial visualization, each bar appears in a circle with longer bars that represent larger values. Hover over a bar to see the details about it, such as the exact value represented by the bar. Each bar starts at 12 noon and goes in a clockwise direction for positive values and counterclockwise for negative values.

Radial visualizations, also known as dial charts or speedometer charts, show information as reading on a dial. The radial visualization is valid only with one category.

For example, this visualization shows renewals by offer type and gender.

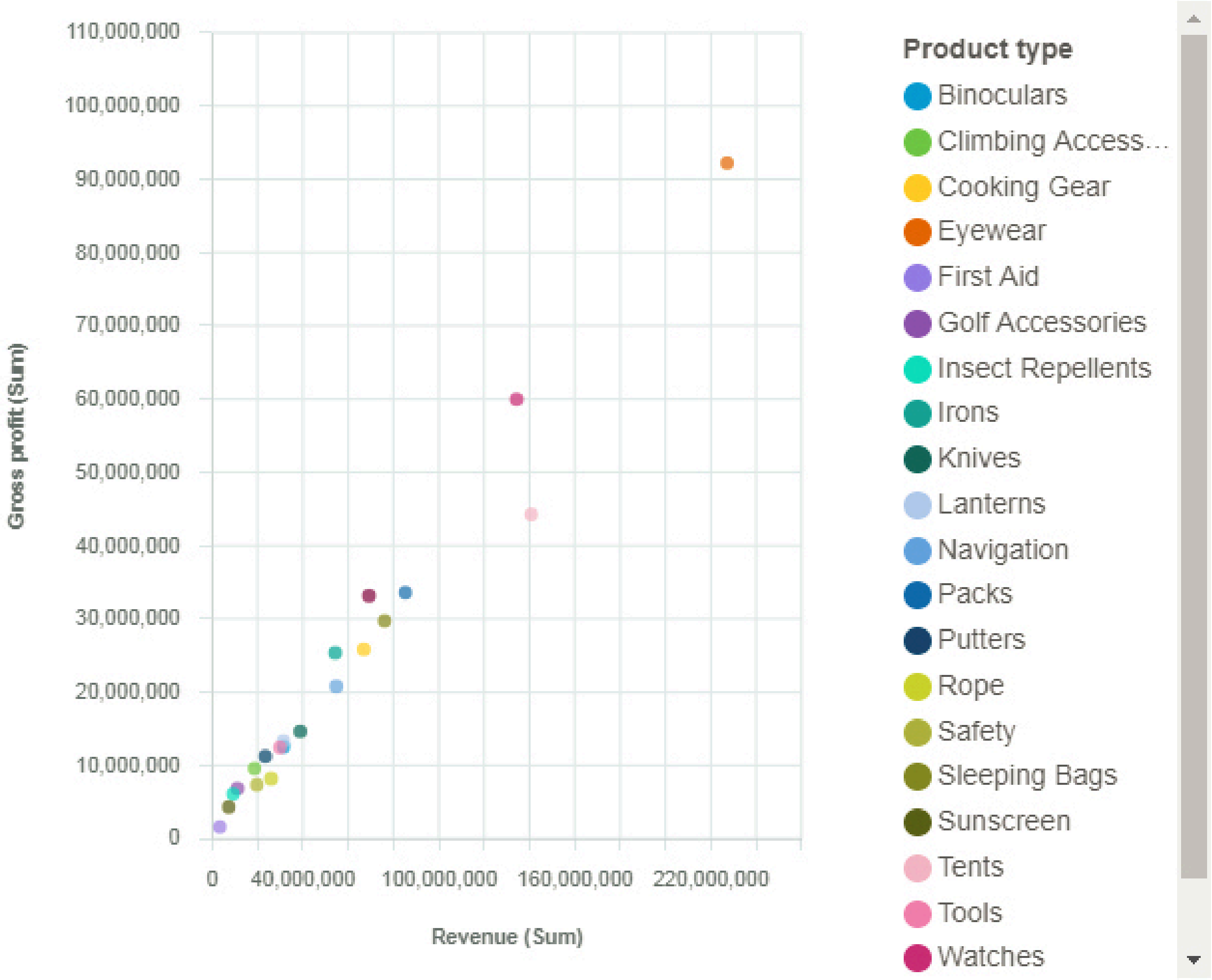


Scatter

Scatter visualizations use data points to plot two measures anywhere along a scale, not only at regular tick marks.

Scatter visualizations are useful for exploring correlations between different sets of data.

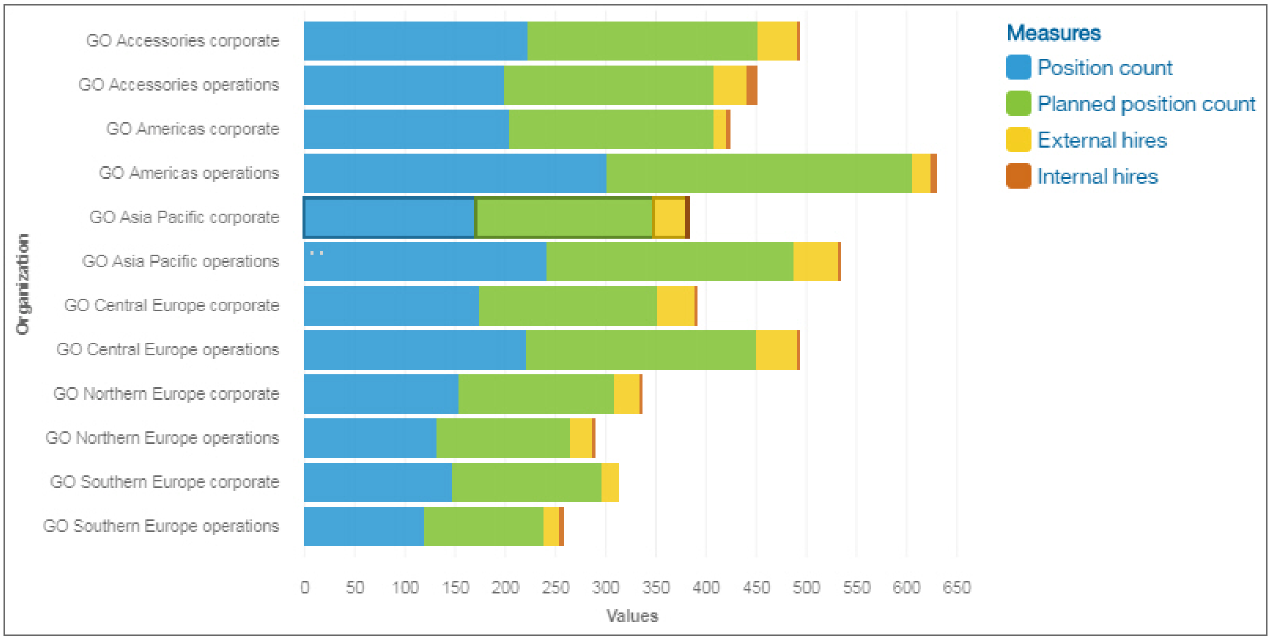
The following example shows the correlation between revenue and gross profit for each product type.



Stacked bar

Use a stacked bar visualization to compare the proportional contributions for each item to the total, such as sales for products and sales for products each month.

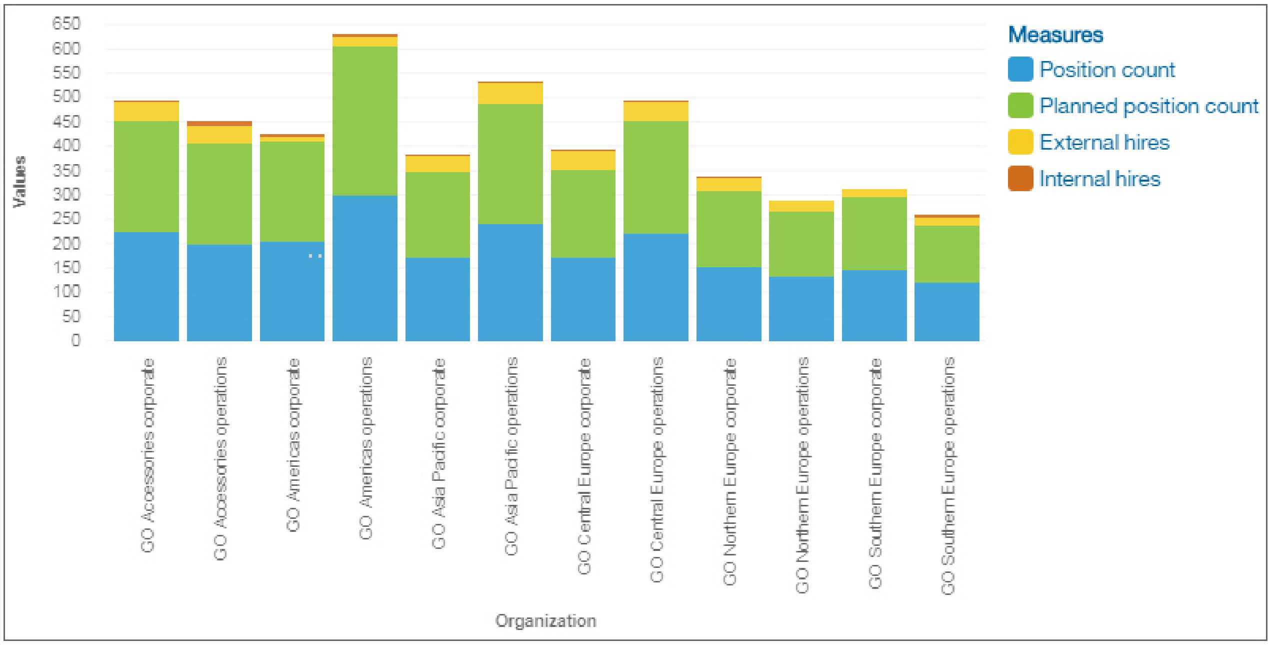
A stacked bar visualization can show change over a specific time period or compare the proportional contributions for each item to the total. If there are so many bars that the labels are impossible to read, filter the data to focus on a subset of the data or use a tree map.



Stacked column

Use a stacked column visualization to compare the proportional contributions for each item to the total, such as sales for products and sales for products each month.

A stacked column visualization can show change over a specific time period or can compare the proportional contributions for each item to the total. If there are so many bars that the labels are impossible to read, filter the data to focus on a subset of the data or use a tree map.



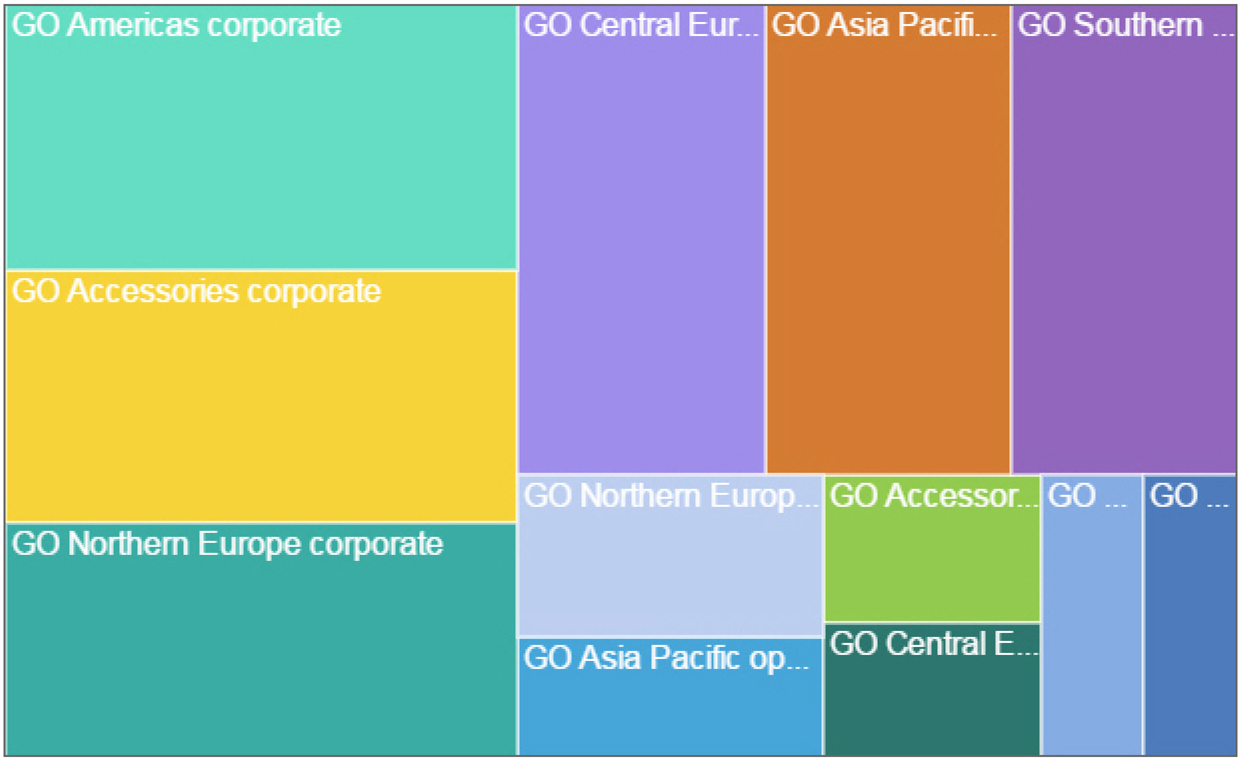
Treemap

Use a tree map visualization to identify patterns and exceptions in a large, complex data asset.

Treemaps show relationships among large numbers of components by using size and color coding in a set of nested rectangles.

A treemap that is colored by category identifies the level 1 category by color. The sizes of the rectangles represent the values. In a treemap that is colored by value, the sizes of the rectangles represent one of the values and the color represents a second set of values. Do not use data that includes negative numbers. A treemap ignores negative numbers.

For example, this treemap visualization shows course cost by organizational unit.



Waterfall

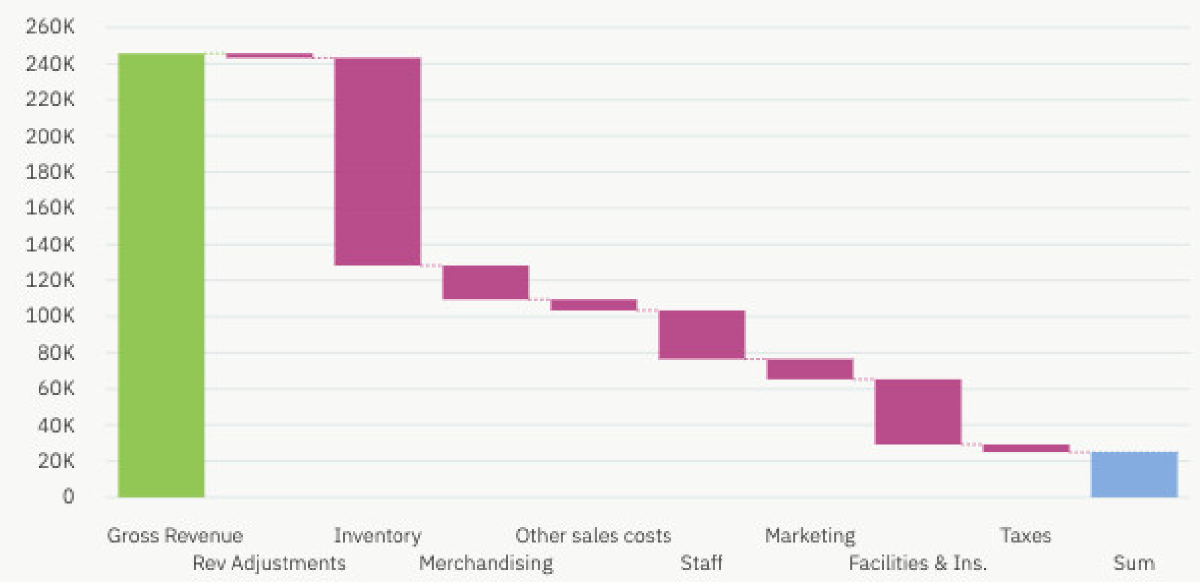
Use a waterfall visualization to understand the cumulative effect a series of positive and negative values have on an initial value. The bars in a waterfall visualization are not totals.

A waterfall visualization shows how an initial value is increased and decreased by a series of intermediate values, leading to a final cumulative value shown in the far right column. The intermediate values can either be time-based or category-based.

Some examples of waterfall visualizations are as follows:

- Viewing the net income after you add the increases and decreases of revenue and costs for an enterprise over a quarter.
- Cumulative sales for products across a year with an annual total.

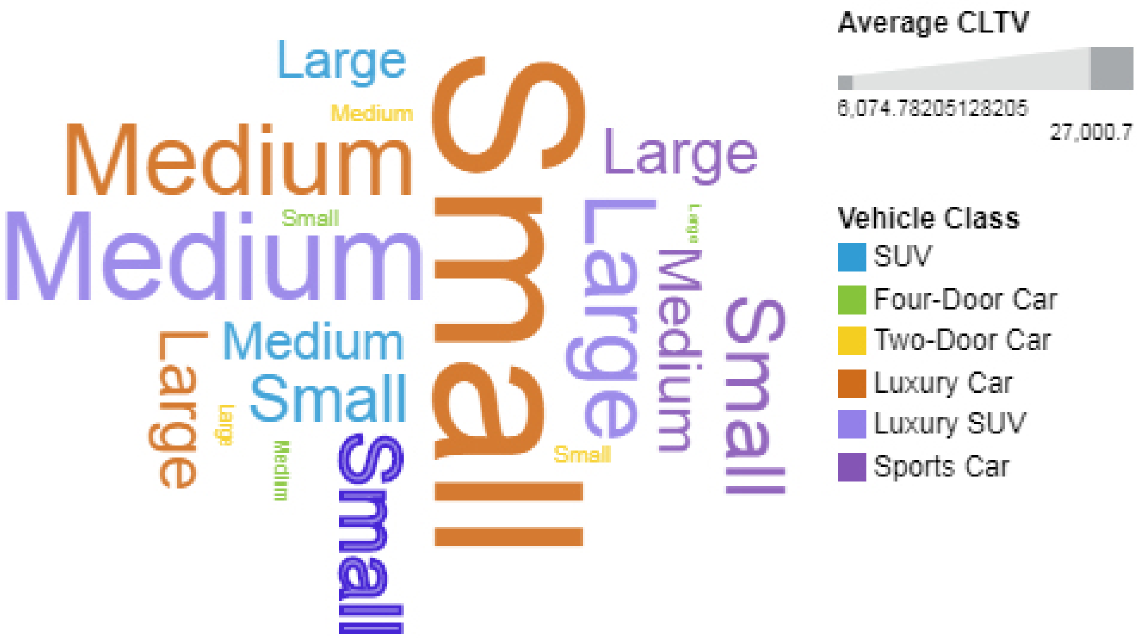
This waterfall visualization shows the policy holder delta by month.



Word cloud
Use a word cloud visualization when you want to see a text-based visualization of a column. The text height represents the scale. The name itself is the different members of the column.

Tip: The view should contain at least 15 columns and at least 100 rows to create an effective word cloud.

For example, this word cloud visualization shows the customer life time value by vehicle size and class.



Visualizations in Planning Analytics Workspace Classic

You can use any of the visualizations that are described here to present data in Planning Analytics Workspace Classic.

View this video to learn how to build visualizations in Planning Analytics Workspace Classic.

https://youtu.be/ImyJxUHnjSc

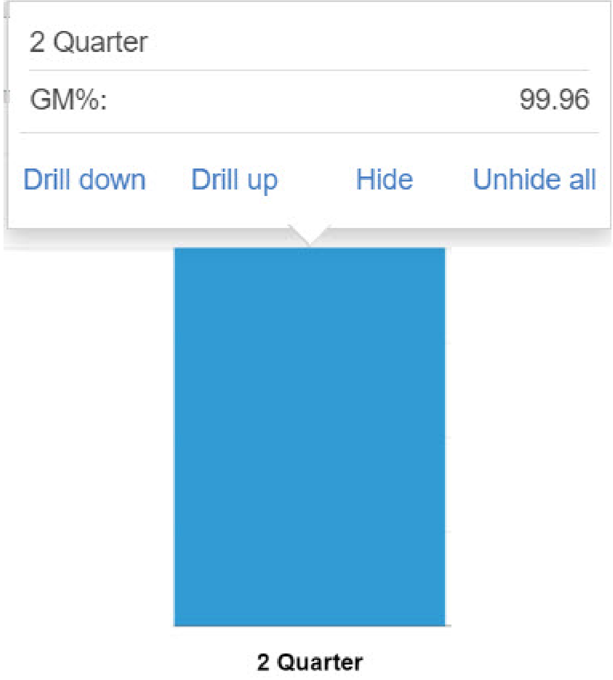
To change the way data is presented, click a view, and then click . Then, click the **Change visualization** icon .

The visualizations available vary according to the dimensionality and configuration of your view. The **Change visualization** list contains only the visualizations that can be rendered with your current view configuration. For an accounting of which visualizations are available for specific view configurations, see "Visualizations available by view configuration" on page 85.

When you create a visualization, columns are always used as measures, while members on the row dimensions are visualization categories.

A visualization displays only as many columns of data as the number of measures it supports. For example, a Stack Bar (1 measure) visualization displays only the first column of data and ignores all other columns. Similarly, a Line and Column (2 measure) visualization displays only the first two columns of data and ignores any remaining columns. Details on the number of measures that are supported for specific view configurations can be found in "Visualizations available by view configuration" on page 85.

If you click on a visualization member, you'll see several options to interact with the visualization.



2 Quarter

GM%: 99.96

Drill down
Reveal the members of a consolidation. You cannot drill down on a leaf member.

Drill up
Collapse the member to show its immediate parent.

Hide
Hide the member in the visualization.

Unhide all
Reveal all hidden members in the visualization.

The following visualizations are available in Planning Analytics Workspace:

Area
An area visualization emphasizes the magnitude of change over time.

Because an area visualization stacks the results for each column or item, the total of all results is easily seen.

For example, an area visualization is excellent for looking at revenue over time across several products.

Bar

A bar visualization uses horizontal bars to show the values in individual groups or categories. The length of a bar indicates each value. Bar visualizations are useful for comparing values.

For example, a bar visualization might show the number of males and females who purchased a specific item. The length of one bar would show the number of males, and the length of the other bar would show the number of females. By checking the length of the bars, you can easily compare the values in the groups or categories.

Bubble

A bubble visualization shows relationships among columns that contain numeric values, such as revenue and profit.

The bubbles are in different sizes and colors. The x-axis represents one measure, the y-axis represents another measure, and the size of the bubbles represents the third measure.

For example, a bubble visualization shows cost in the x-axis, revenue in the y-axis, and quantity sold for all products. There is one bubble for each product. The location of the bubble in the visualization indicates the product's cost and revenue. The size of the bubble indicates the quantity sold.

Because a bubble visualization uses area to represent numbers, it is best for positive values. If your data set includes negative values, they are shown in a different color: a circle for 100 and a circle for -100 will both be the same size, but 100 might be blue and -100 might be red. If your data set has many negative numbers, consider that uses a bar visualization.

Column

A column visualization uses vertical bars to show the values in individual groups or categories. The height of a bar indicates each value. Column visualizations are useful for comparing values.

For example, a column visualization might show the number of car models sold in a region. The height of one bar would show the number of one car model, and the height of another bar would show the number of a different model. By checking the height of the bars, you can easily compare the values in the groups or categories.

Exploration

An Exploration shows data in rows and columns using a grid-style layout.

Heat

A heat map visualization shows the relationship between columns, using color and intensity.

Line

A line visualization shows trends over time.

A line visualization can compare trends and cycles, infer relationships between variables, or show how a single variable is performing over time.

For an effective line visualization, the x-axis should show time, such as years, quarters, months, or days. If the x-axis shows something else, such as individual countries, use a bar visualization instead.

Line and column

Display requirements: a defined measures dimension with two members.

A line and column visualization shows values for two measures, with one measure represented by columns and the other measure represented with a line.

List

A list visualization displays values in a list ordered first by row members, and then by column members.

Map

Display requirements: a defined geography dimension on the row axis of your view.

A map visualization shows patterns in your data by geography.

Your data set must contain geographical data, such as countries, states, or provinces. To determine if a dimension is mappable, Planning Analytics Workspace analyzes a sample of 2000 values in the row dimension, looking for recognizable place names. If 80% or more of the members in the geography dimension are recognized as map values, a map is generated.

For example, you have four countries in your geography dimension: Brazil, China, Indai, and Russia. The misspelling of India means that only 75% of the values are recognizable place names and you cannot generate a map.

For more information on the languages and geographic entities supported in map visualizations, see "Map reference info" on page 647.

For details on configuring a view to display a map visualization, see "Create a map visualization" on page 88.

Packed bubble

A packed bubble visualization shows relationships among columns that contain numeric values, such as revenue. It's a good choice when you want to display a large amount of data in a small space.

The bubbles are in different sizes and colors.

Because a packed bubble visualization uses area to represent numbers, it is best for positive values. If your data includes negative values, they display in a different color: a circle for 100 and a circle for -100 are both the same size, but 100 might be blue and -100 might be red. If your data has many negative numbers, consider using a bar visualization.

Pie

A pie visualization displays values as segments of a circle, or as slices of a pie.

Point

A point visualization uses multiple points to show trends over time. It is similar to a line chart, but without the lines; only the data points are shown.

Radial

A radial visualization displays values as segments of a single ring. The length of a segment in the ring indicates value.

Radial bar

A radial bar visualization displays values as concentric rings of a circle. It's similar to a standard bar chart, but the bars are bent into a circular shape.

Stack bar

A stack bar visualization is similar to a regular bar visualization, but instead of grouping values next to each other and displaying individual bars, values are placed in a single bar and positioned end-to-end. The length of a segment in the bar indicates value.

Stack column

A stack column visualization is similar to a regular column visualization, but instead of grouping values side-by-side and displaying individual columns, values are placed in a single column and positioned on top of each other. The height of a segment in the column indicates value.

Tree map

A tree map visualization identifies patterns and exceptions in a large, complex data set.

Tree maps show relationships among large numbers of components by using size and color coding in a set of nested rectangles.

A tree map that is colored by category identifies the level 1 category by color. The sizes of the rectangles represent the values. In a tree map that is colored by value, the sizes of the rectangles represent one of the values and the color represents a second set of values. Do not use data that includes negative numbers. A tree map ignores negative numbers.

Word cloud

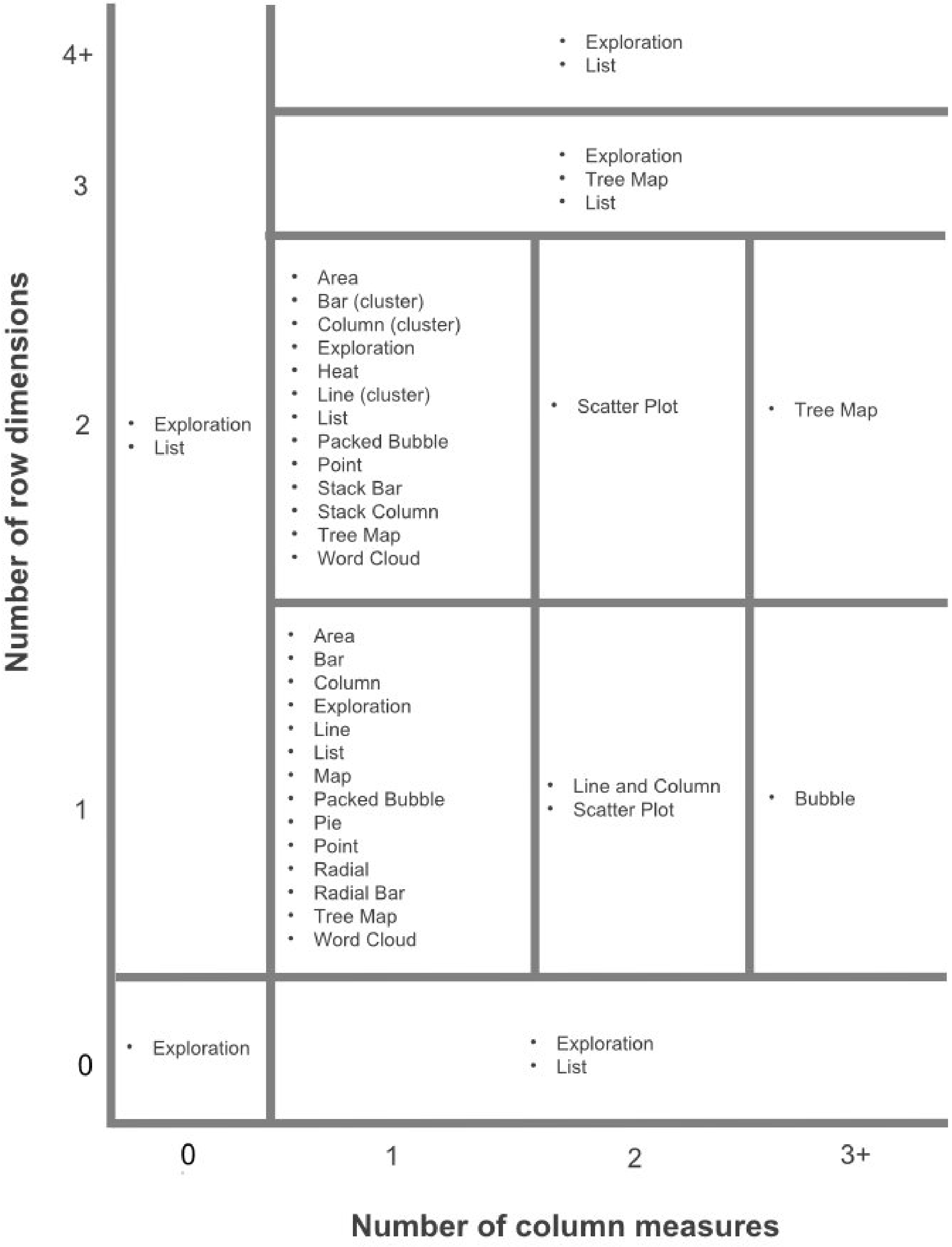
A word cloud visualization presents a visual representation of text values. The more frequently a text string occurs in your data, the larger the string appears in the word cloud.

Visualizations available by view configuration

The visualizations that are available when you click the **Change visualization** icon vary according to your view configuration. Only the visualizations that can be displayed for the current view configuration are shown in the **Change visualization** list.

The number of stacked dimensions on the row axis and the number of columns to be used as measures determine which visualizations can be displayed. For a visualization that supports a single measure, only the first column in your view is used as the measure, even if there are multiple columns present. For a visualization that supports two measures, only the first two columns are used, and so on.

The following image illustrates which visualizations can be displayed, based on the number of column members and row dimensions.



Number of row dimensions	0	1	2	3+
4+	• Exploration • List	• Exploration • List		
3		• Exploration • Tree Map • List		
2		• Area • Bar (cluster) • Column (cluster) • Exploration • Heat • Line (cluster) • List • Packed Bubble • Point • Stack Bar • Stack Column • Tree Map • Word Cloud	• Scatter Plot	• Tree Map
1		• Area • Bar • Column • Exploration • Line • List • Map • Packed Bubble • Pie • Point • Radial • Radial Bar • Tree Map • Word Cloud	• Line and Column • Scatter Plot	• Bubble
0	• Exploration	• Exploration • List		

Note: If the first column in the query contains text cells, it will be treated as zero for single measure charts and will display zeros. To resolve it, you will need to edit the set and remove the column.

Visualizations that can be displayed with no row dimensions
If your view contains no row dimensions, you can display the following visualization, listed by the number of column members present in your view.

No column members
Exploration

One or more column members
Exploration

List

Visualizations that can be displayed with one row dimension
If your view contains one row dimension, you can display the following visualization, listed by the number of column members present in your view.

No column members
Exploration

List

One column member
Exploration

List

Area (1 measure)

Bar (1 measure)

Column (1 measure)

Line (1 measure)

Map (1 measure)

Packed Bubble (1 measure)

Pie (1 measure)

Point (1 measure)

Radial (1 measure)

Radial Bar (1 measure)

Tree Map (1 measure)

Word Cloud (1 measure)

Two column members
Exploration

List

Line and Column (2 measure)

Map (2 measure)

Scatter Plot (2 measure)

Tree Map (2 measures)

Word Cloud (2 measures)

Area (1 measure)

Bar (1 measure)

Column (1 measure)

Line (1 measure)

Packed Bubble (1 measure)

Pie (1 measure)

Point (1 measure)

Radial (1 measure)

Radial Bar (1 measure)

Three or more column members

Exploration

List

Bubble (3 measure)

Map (3 measure)

Radial (2 measure)

Line and Column (2 measure)

Scatter Plot (2 measure)

Tree Map (2 measure)

Word Cloud (2 measure)

Area (1 measure)

Bar (1 measure)

Column (1 measure)

Line (1 measure)

Packed Bubble (1 measure)

Pie (1 measure)

Point (1 measure)

Radial Bar (1 measure)

Visualizations that can be displayed with two stacked row dimensions

If your view contains teo stacked row dimensions, you can display the following visualization, listed by the number of column members present in your view.

No column members

Exploration

List

One column member

Exploration

List

Area (1 measure)

Bar (cluster, 1 measure)

Column (cluster, 1 measure)

Heat (1 measure)

Line (cluster, 1 measure)

Packed Bubble (1 measure)

Point (1 measure)

Area (1 measure)

Stack Bar (1 measure)

Stack Column (1 measure)

Tree Map (1 measure)

Word Cloud (1 measure)

Two or more column members

Exploration

List

Scatter Plot (2 measure)

Tree Map (2 measure)

Area (1 measure)

Bar (cluster, 1 measure)

Column (cluster, 1 measure)

Heat (1 measure)

Line (cluster, 1 measure)

Packed Bubble (1 measure)

Point (1 measure)

Stack Bar (1 measure)

Stack Column (1 measure)

Word Cloud (1 measure)

Visualizations that can be displayed with three stacked row dimensions
If your view contains three stacked row dimensions, you can display the following visualization, listed by the number of column members present in your view.

No column members

Exploration

List

One column member

Exploration

List

Tree Map (1 measure)

Two or more column members

Exploration

List

Tree Map (2 measures)

Visualizations that can be displayed with four or more stacked row dimensions
When your view contains four or more stacked row dimension, the only visualizations available are Exploration and List, regardless of the number of column members.

Create a map visualization

A map visualization shows patterns in your data by geography.

About this task

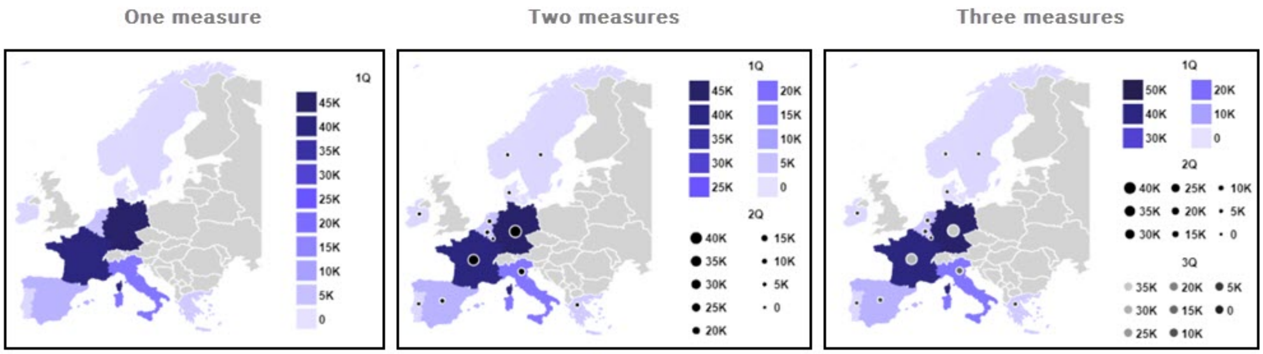
To create a map visualization, your view must contain a defined geography dimension, and that dimension must be on the row axis of your view.

The geography dimension must contain elements that are recognizable as geographic entities. To determine if a dimension is mappable, Planning Analytics Workspace analyzes a sample of members on the row dimension, looking for recognizable place names. If 80% or more of the members in the geography dimension are recognized as geographic entities, a map is generated.

For more information on the languages and geographic entities supported in map visualizations, see "Map reference info" on page 647.

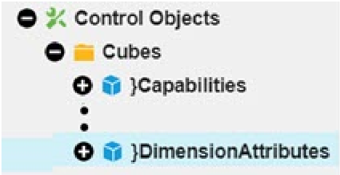
A map visualization can display up to three measures.

- A one-measure map shades the regions based on the measure value. To display a one-measure map, there must be a single measure on the column axis.
- A two-measure map shades the regions based on the first measure, and shows bubbles on each region sized by the second measure. To display a two-measure map, there must be only two measures on the column axis.
- A three-measure map shades the regions based on the first measure, shows bubbles on each region sized by the second measure, and shades those bubbles based on the third measure. To display a three-measure map, there can be any number of measures on the column axis, but only the first three measures are included in the map.



Procedure

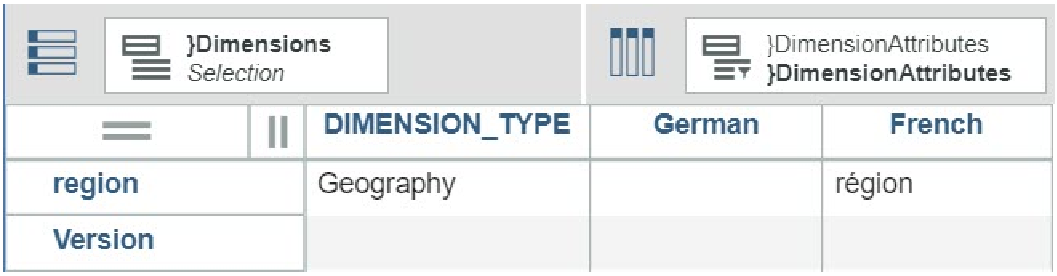
1. Define the dimension that contains geographic elements as a geography dimension.
   a) In the content tree, locate and open the }DimensionAttributes control cube.



Control Objects
Cubes
}Capabilities
}DimensionAttributes

   b) At the intersection of the dimension that you want to define as a geography dimension and the DIMENSION_TYPE attribute, enter `Geography`.

   In this example, the region dimension is defined as a geography dimension.



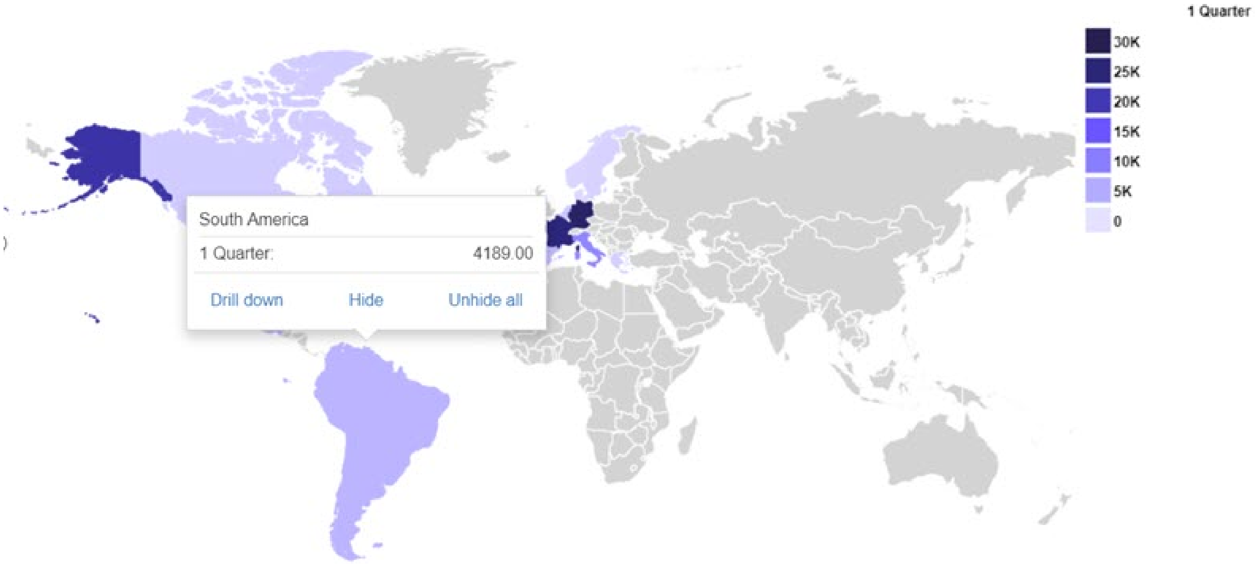
}Dimensions
Selection
}DimensionAttributes
}DimensionAttributes

	DIMENSION_TYPE	German	French
region	Geography		région
Version			

2. Create a view with the defined geography dimension as the sole dimension on the row axis.
3. Click anywhere on the view and then click .
4. Click the **Change visualization** icon and then click **Map** .

Results

Your view is rendered as a map. You can click on any member in the map to reveal exact values and to interact with the map.



- **Drill down** - Expose underlying details for the member.
- **Hide** - Exclude the member from the visualization.
- **Unhide all** - Restore any members that were previously hidden in the visualization.

Customize visualizations

You can customize your view of data in a visualization.

Drill down to reveal detail

To reveal underlying detail, first click a part of the visualization, then click **Drill down**.

You can drill down on the following parts of a visualization:

- a legend member label
- an axis member label
- a data point in the visualization. For example, an individual segment in a stacked bar visualization, a bubble in a packed bubble visualization, or a cell in a heat visualization.

When you drill down on a data point in a visualization, and there are consolidations on both axes, the drill down occurs on both axes.

There is no roll up option. To reverse a drill down action, click the **Undo** icon ↺

Hide members

Click a legend member label or axis member label, then click **Hide** to hide the member in your visualization.

Unhide members

You cannot selectively unhide individual members, you must unhide all hidden members.

To unhide members, click any legend member label or axis member label, then click **Unhide all**.

Set visualization properties

You can set properties that determine the appearance of a visualization.

About this task

Setting properties to determine the appearance of a visualization can be useful. For example, you can pick the color palette or set the position of the legend in a visualization. The properties that you can set vary according to the type of visualization you are using; not all properties are available for all visualizations.

Procedure

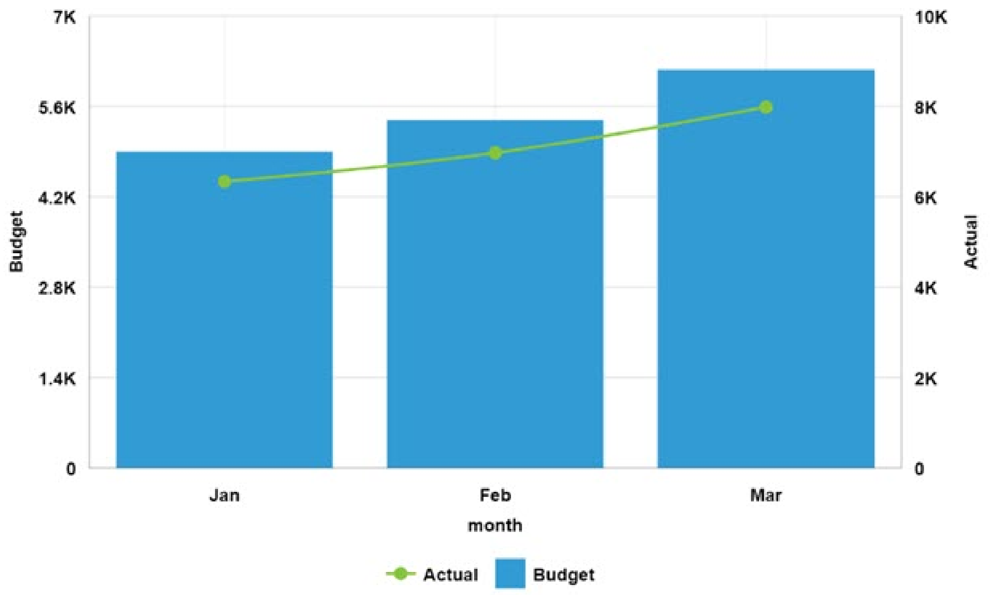
1. Ensure that you are in Edit mode.
2. Click anywhere on the visualization.
3. Click the **Properties** icon , then **Visualization details**.
4. Set the properties for your visualization.
   - **Color palettes** - Pick the color palette that you want to see in the visualization.
   - **Show summary values in chart** - Enable to include consolidations in the visualization. Disable to see only leaf values in the visualization.
   - **Share vertical axis range** - Applicable only to Line and Column visualizations. By default, Line and Column visualizations use separate vertical axis value ranges for the two measures in a visualization. You can enable the **Share vertical axis range** property to use a common vertical axis range for both measures, resulting in a more accurate visual representation of your data. For a full explanation of the impact of this property, see "Use the Share vertical axis range property in Line and Column visualizations" on page 91.
   - **Hide grid lines** - Enable to show grid lines in the visualization. Disable to hide grid lines.
   - **Hide axis title labels** - Enable to hide axis titles. Disable to make axis titles visible. Axis titles often reflect the dimensions being charted, but in many visualizations one of the axis titles is `Value`.
   - **Hide leaf labels** - Applicable only to Tree Map visualizations. Enable to hide labels on leaf members in the visualization, which can be useful when a visualization contains many leaf members. Disable to display labels for leaf members.
   - **Abbreviate numbers** - Enable to abbreviate numbers on the value axis or when the legend represents values. Disable to use full numbers. When you abbreviate numbers, "K" is used to represent thousands and "M" represents millions. For example, when **Abbreviate numbers** is enabled, 20,000 appears as 20K, while 15,500,000 appears as 15.5M.
   - **Hide legend** - Enable to hide the legend in the visualization. Disable to show the legend.
   - **Legend position** - Pick the position for the legend.

Use the Share vertical axis range property in Line and Column visualizations

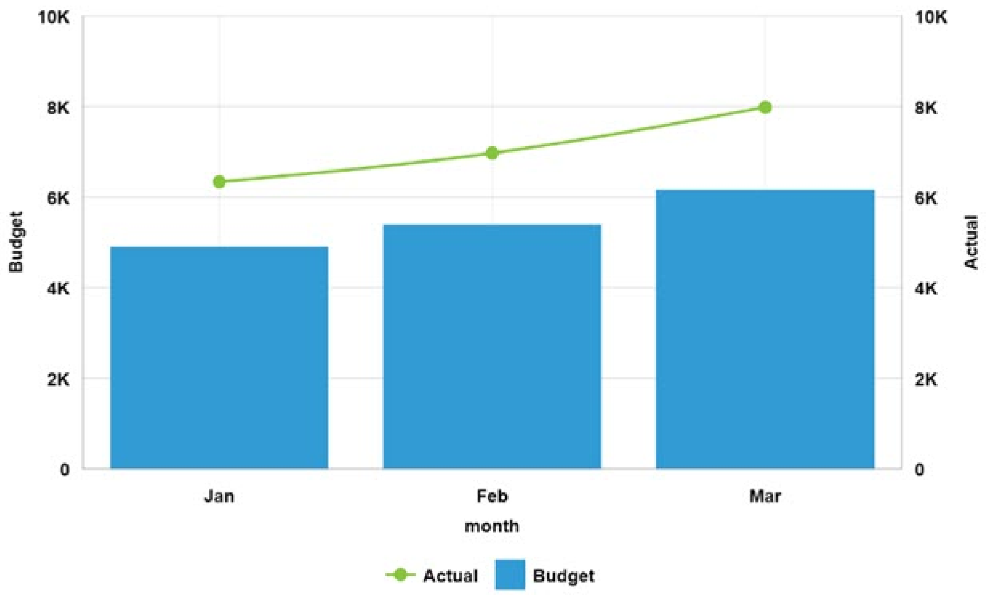
By default, Line and Column visualizations use separate vertical axis value ranges for the two measures in a visualization. In some circumstances when your visualization contains two similar measures, this can result in a visualization that, while accurate, does not provide a clear visual depiction of your data.

About this task

As an example, this visualization shows Actual (green line) and Budget (blue columns) measure values across three months. The Budget vertical axis range is 0 - 7,000, while the Actual vertical axis range is 0 - 10,000. The visualization accurately portrays Actual and Budget values; for any given month, the Actual values are greater than the Budget values. However, a quick glance at the visualization implies that the Budget values exceed Actual values, because the vertical axis value ranges differ. The chart is technically accurate, but visually confusing.



You can alleviate this confusion by using the **Share vertical axis range** property. When you enable the **Share vertical axis range** property for the visualization, the Actual and Budget measures use the same vertical axis value range of 0 - 10,000. The result is a visualization that clearly shows Actual values exceeding Budget values in every month.



Procedure

1. In edit mode, click anywhere on the Line and Column visualization.
2. Click the **Properties** icon, then **Visualization details**.
3. Enable the **Share vertical axis range** property.

Forecasting

Use forecasting in IBM Planning Analytics Workspace to discover and model trend, seasonality, and time dependence in data.

You can forecast in Planning Analytics Workspace by using automated tools that model time-dependent data. Automated model selection and tuning makes forecasting easy to use, even if you are not familiar with time series modeling.

Forecasts and their confidence bounds are displayed in visualizations as a continuation of historic data. You can also view the statistical details for generated models if you want to see the technical background.

Specifying time series in forecasts often requires data manipulation. Planning Analytics Workspace supports a wide range of time series without the need for manipulation, ranging from standard date and time types, to nested periodic and cyclical time fields. When data is recognized as a time series, data preparation is automated. Appropriate trend and seasonal periods are detected, and models are selected from a set of nine different model types.

You can forecast in line, bar, and column visualizations. Forecasts and confidence bounds are computed for each time series, and displayed in the visualization as extensions of the current data. You can inspect each time series separately, and tailor the forecast and results to your own data and requirements.

If you are familiar with forecasting models, you can view the selected model type, estimated model parameters, standard accuracy measures, and processing summary information.

Forecasting features

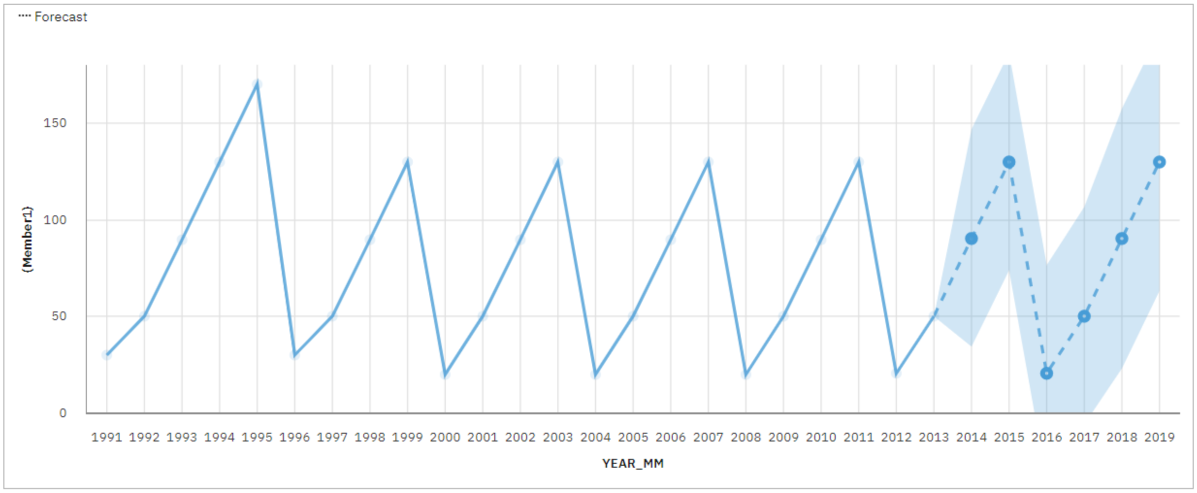
Forecasting is started from an exploratory view.

When you can use forecasting in your view, a **Forecast** dialog box 📈 is available, where you can modify model and forecast settings, and confidence bounds.

Forecasting requires the time dimension to be placed on the column. The algorithm assumes that the time dimension has no gaps. Forecasting also supports hierarchical time dimensions and nested dimensions as time (that is: year, quarter, and month dimensions). Crosstabs views with dimensions that no represent time on the column can be forecasted, but the process might fail or the results might be unreliable.

Note: When the members of the dimensions, used to indicate the time (columns), have multiple parents, then the forecasting fails. With multiple parents, the correct order of those members is not known. To solve this limitation, create an alternative hierarchy for the dimensions that exclude the extra parents.

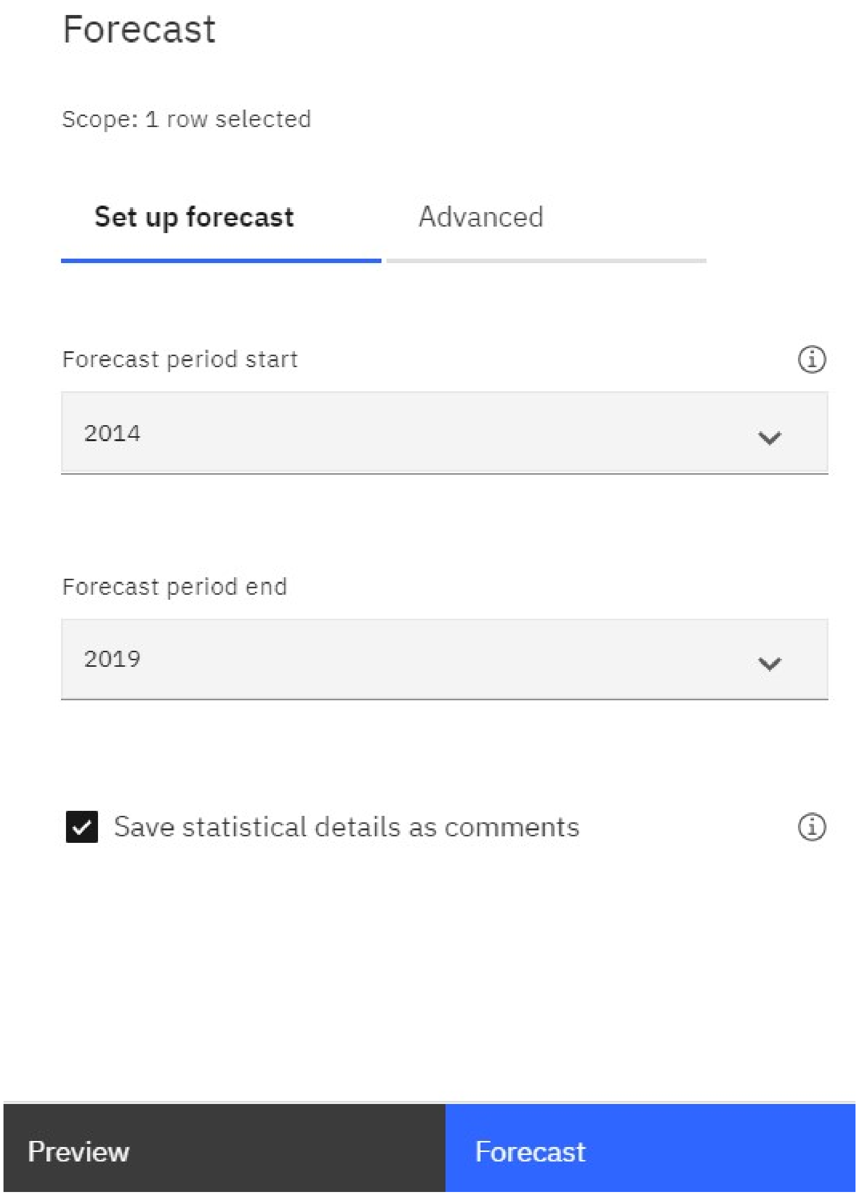
The following example shows forecasting values and confidence bounds in a line visualization.



Forecasting options

You can modify your forecasts by setting a number of period and confidence level options in the **Forecast** dialog box 📈.

The Set up forecast tab



Forecast

Scope: 1 row selected

Forecast period start: 2014

Forecast period end: 2019

☑ Save statistical details as comments

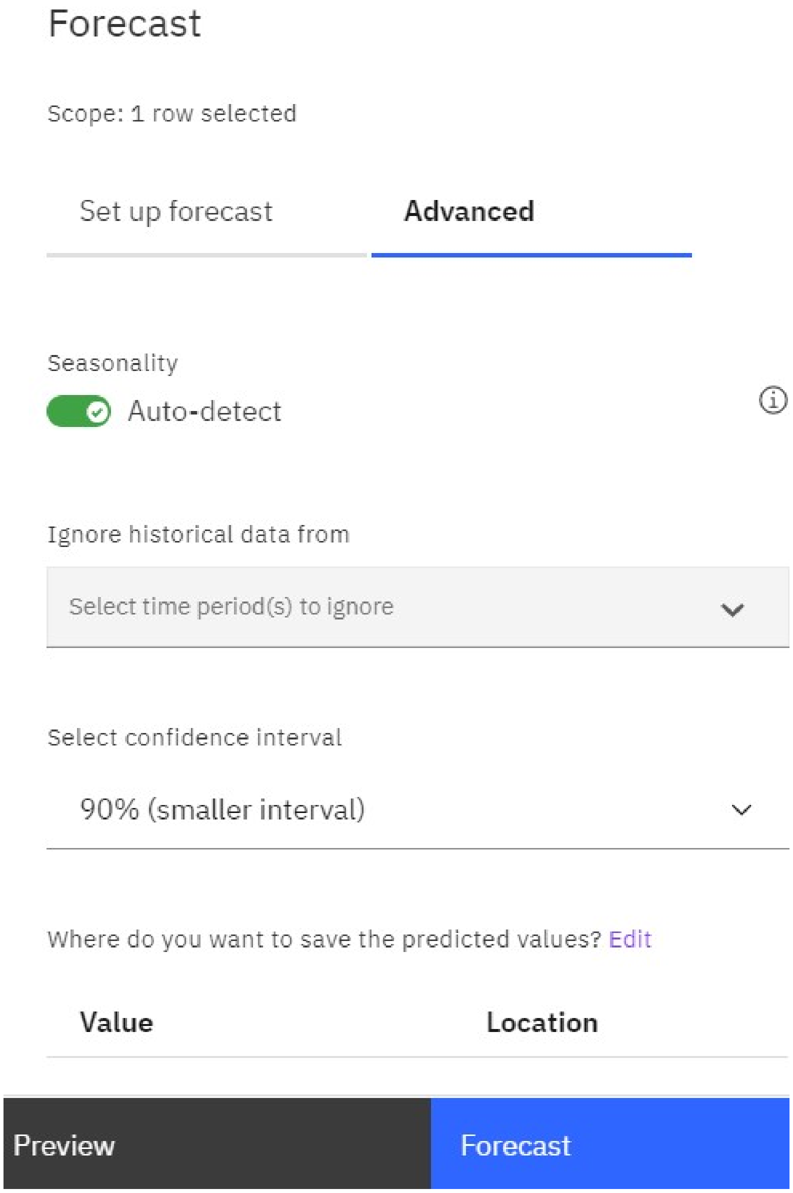
The following options are available on the **Set up forecast** tab.

Forecast period start
The start of the forecast period. A period is the smallest time interval between neighboring points in the data.

Forecast period end
The end of the forecast period. The forecast period end field is populated automatically to reflect the last member on the dimension.

Save statistical details as comments
For performance reasons, the option to save statistical details as comments is only available if the scope of the forecast is less than 25 time series. For more than 25 time series, the option is disabled.

The Advanced tab



Scope: 1 row selected

Seasonality

Auto-detect: on

Ignore historical data from: Select time period(s) to ignore

Select confidence interval: 90% (smaller interval)

Where do you want to save the predicted values? Edit

Value	Location

The following options are available on the **Advanced** tab.

Seasonality

The seasonality with which to build the model. Seasonality is when the time series has a predictable cyclic variation. For example, during a holiday period each year.

The default value is **Auto-detect**. **Auto-detect** automatically detects seasonality by building multiple models with different seasonal periods and choosing the best one.

If you switch **Auto-detect** off, you can specify seasonality by entering a non-negative integer, such as 0, 1, 2, 3 in the **Enter seasonality interval** field.

To specify a non-seasonal model, set the **Enter seasonality interval** to 0 or 1. A model with user specified seasonality is displayed only if the seasonal model is more accurate than all of the non-seasonal models.

Ignore historical data from

Ignores a specified number of data points at the end of a time series when building the model and computing the forecasts. Any missing values at the end of a non-ignored portion of a series is also forecast. Ignored last periods value must be specified selected from the menu that lists the time dimension.

By default, no historical data is ignored. If no missing values exist, then all of the historical data is used and the first forecast point is after the last historical data point.

Ignoring the last data period can be useful when the data is incomplete. For example, you might be doing a forecast halfway through a month. Exclude this month from the forecast by setting **Ignored historical data from** to the last data point.

Select confidence interval

The certainty with which the true value is expected to be within the specified range. You can see corresponding confidence interval in a tooltip by hovering over any forecast value. The confidence interval is displayed as higher and lower bounds.

You can select three different confidence levels: 90%, 95%, and 99%. The default is 95% and the lower and higher bound define the range at which you can be 95% confident that the true value lies within that range.

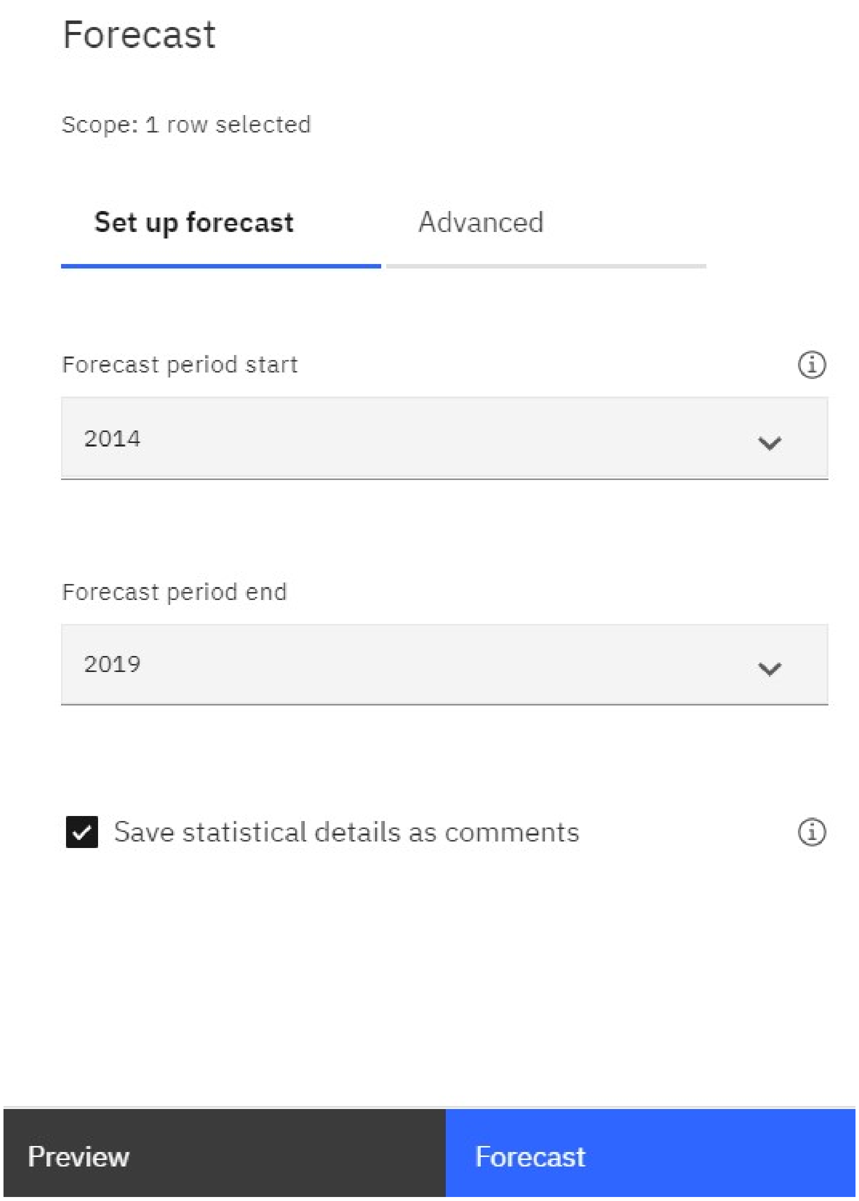
Previewing your forecast

When the forecast options are set up, you can preview the forecast.

Before you begin

Note: The preview operation runs on sandboxes and does not change your actual data.

You can preview only one time series at a time. The forecasting preview shows what the results for the forecast look like, based on your input. By default **Preview** is not active. Select one row and you are able to click **Preview**.


Forecast

Scope: 1 row selected

Forecast period start: 2014

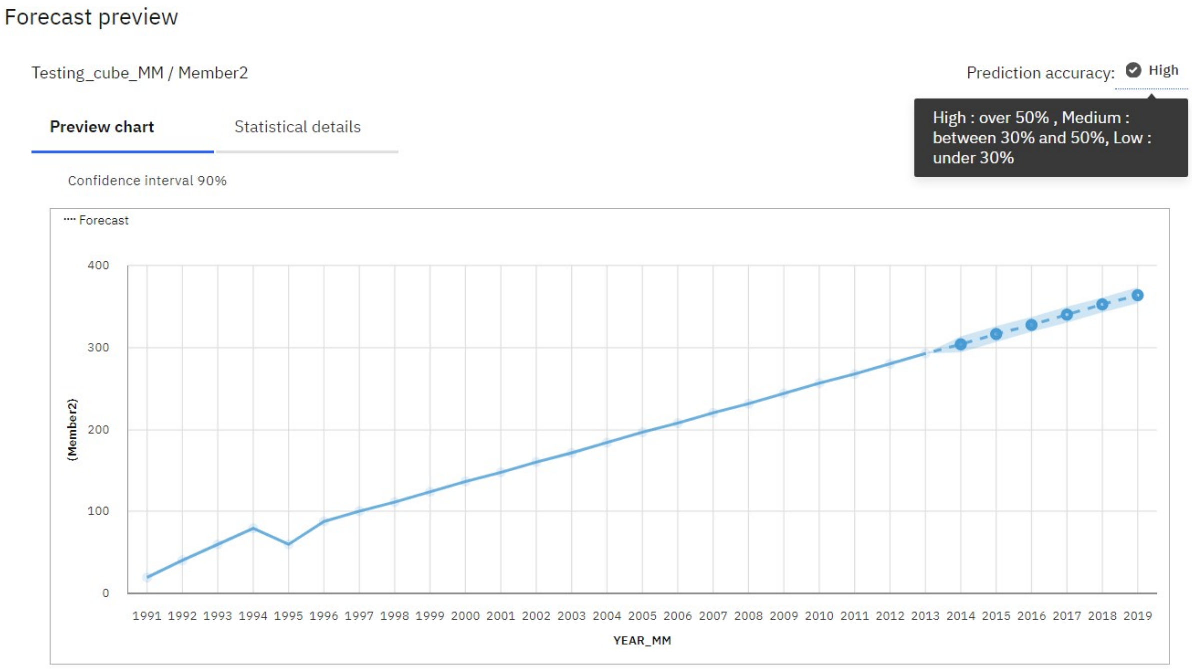
Forecast period end: 2019

☑ Save statistical details as comments

Procedure

1. On the **Forecast** dialog box, click **Preview**.

   The preview is displayed. A visualization displays the forecasted data along with the high and low ranges. The **Prediction accuracy** shows: low, medium, or high.


Forecast preview

Testing_cube_MM / Member2

Prediction accuracy: High

High : over 50% , Medium : between 30% and 50%, Low : under 30%

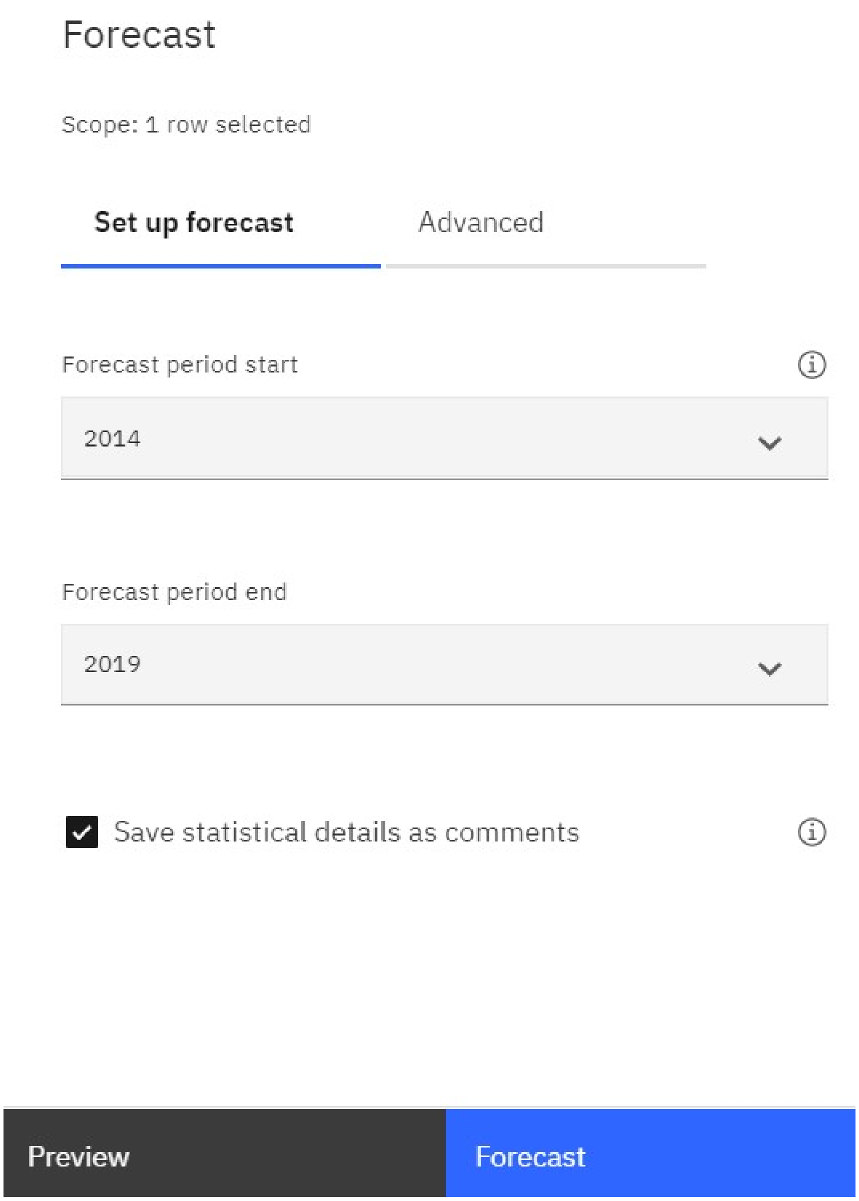
2. The **Statistical details** tab, shows all the details for the forecast.

Forecasting on actuals

When the forecast options are set up and you previewed the forecast, you can forecast on the actuals.

Before you begin

By clicking **Forecast**, you change the TM1 data. The settings in the **Forecast** dialog box are tied to the crosstab used for the process. To share the forecasting parameters with other users, save the book. Different views even the ones that are created from the same TM1 cube do not share parameters.



Forecast

Scope: 1 row selected

Forecast period start: 2014

Forecast period end: 2019

☑ Save statistical details as comments

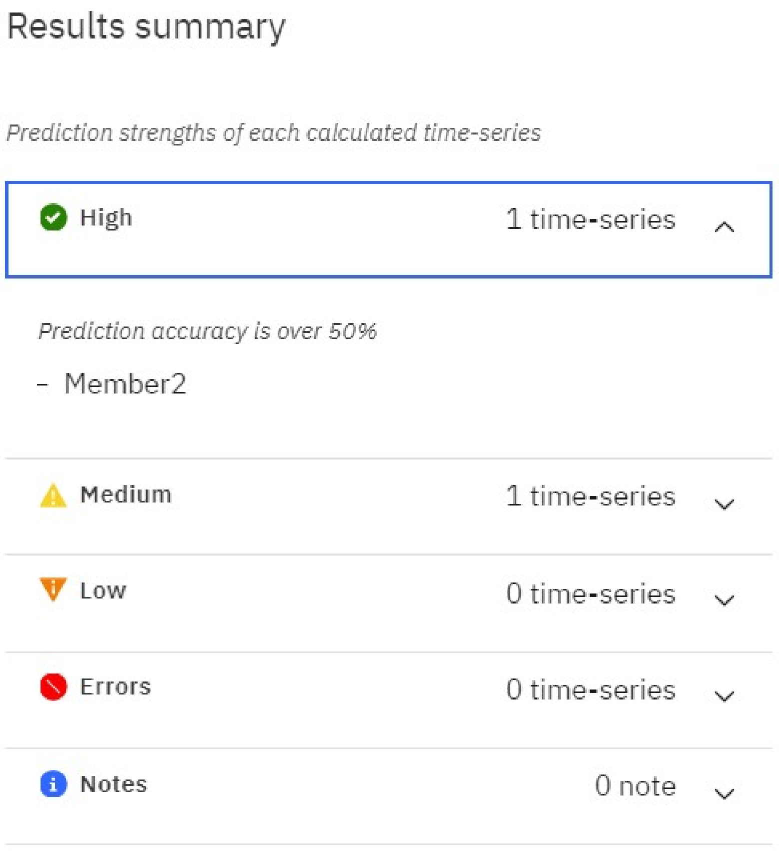
Procedure

1. On the **Forecast** dialog box, click **Forecast**.

   It takes some time for the forecast to complete. When completed, a message is displayed that the AI driven forecast completed successfully.

   The data in the crosstab refreshes.

   The **Result summary** and the **Details** display information on the quality of the forecast. The details specify which time series landed on each quality hierarchy (High, Medium, Low, Errors). The accuracy measure from the forecasting engine defines the hierarchy.



Results summary

Prediction strengths of each calculated time-series

✓ High	1 time-series

Prediction accuracy is over 50%

- Member2

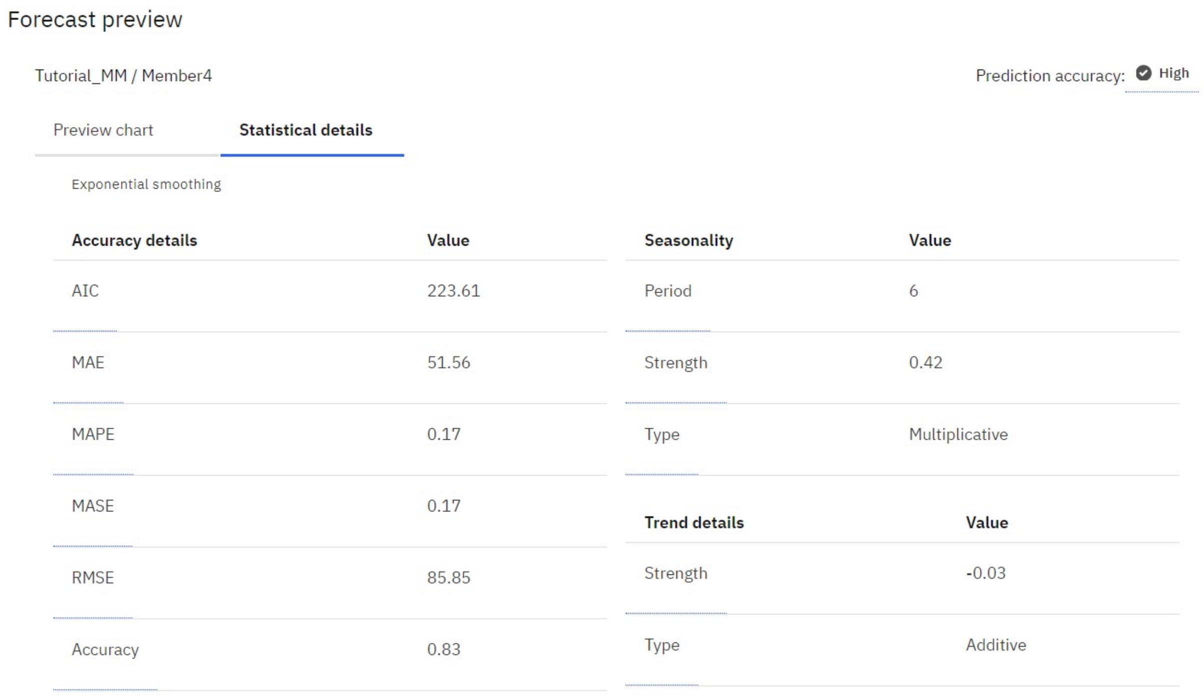
⚠ Medium	1 time-series
▼ Low	0 time-series
⦸ Errors	0 time-series
ℹ Notes	0 note

2. The **Statistical details** tab, shows all the details for the forecast.
3. If you do not want the forecasted date to be displayed in cells in the crosstab, use the option **Save statistical data as comments**. For each time-series forecast row, statistical details are saved as a comment on the first forecast cell. Available for up to 25 rows.

Forecasting statistical details

A forecasting run generates forecasts and forecasting statistical details. Forecasting statistical details are located in the **Statistical details** tab on the **Forecasting preview**.

Forecast detailed information contains forecast **Accuracy details**, **Parameters**, **Seasonality**, and **Accuracy details**.



Forecast preview

Tutorial_MM / Member4

Prediction accuracy: High

Exponential smoothing

Accuracy details	Value
AIC	223.61
MAE	51.56
MAPE	0.17
MASE	0.17
RMSE	85.85
Accuracy	0.83

Seasonality	Value
Period	6
Strength	0.42
Type	Multiplicative

Trend details	Value
Strength	-0.03
Type	Additive

Model information specifies the **Trend** and **Seasonality** type that is selected for estimating the time series data when successful. The following table lists the different available types.

Trend component	**Seasonal component**		
	N NONE	A ADDITIVE	M MULTIPLICATIVE
N NONE	(N, N)	(N, A)	(N, M)
A ADDITIVE	(A, N)	(A, A)	(A, M)
Ad ADDITIVE_DAMPED	(Ad, N)	(Ad, A)	(Ad, M)

Accuracy measures

Model accuracy measures Mean Absolute Error (MAE), Mean Absolute Scaled Error (MASE), Accuracy Percent, Root Mean Squared Error (RMSE), Mean Absolute Percent Error (MAPE), and Akaike Information Criterion (AIC) are based on the time series data that is used to generate the model. All accuracy measures are based on the historical data. Accuracy measures can also be used as an indicator of the forecast accuracy, but they do not carry over to future values.

Mean Absolute Error (MAE)

Computed as the average absolute difference between the values fitted by the model (one-step ahead in-sample forecast), and the observed historical data.

Mean Absolute Scaled Error (MASE)

The error measure that is used for model accuracy. It is the MAE divided by the MAE of the naive model. The naive model is one that predicts the value at time point t as the previous historical value. Scaling by this error means that you can evaluate how good the model is compared to the naive model. If the MASE is greater than 1, then the model is worse than the naive model. The lower the MASE, the better the model is compared to the naive model.

Accuracy Percent (Accuracy %)
The primary indicator of the model accuracy based on the fitted values. It is specified as the reduction percentage of mean absolute error relative to the naïve model. It is computed by subtracting MASE from 1 and expressing it as percentage. If MASE is greater than or equal to 1, the accuracy is set to 0% because the model does not improve upon the naïve model. Higher accuracy indicates lower model error relative to the naïve model.

Mean Squared Error (MSE)
The sum of squared difference between the values that are fitted by the model, and observed values that are divided by the number of historical points, minus the number of parameters in the model. The number of parameters in the model is subtracted from the number of historical points to be consistent with an unbiased model variance estimate.

Root Mean Squared Error (RMSE)
The square root of the MSE. It is on the same scale as the observed data values.

Mean Absolute Percent Error (MAPE)
The average absolute percent difference between the values that are fitted by the model and the observed data values.

Akaike Information Criterion (AIC)
A model selection measure. The AIC penalizes models with many parameters, and so attempts to choose the best model with a preference towards simpler models. The AIC is the sum of the logarithm of non-adjusted MSE multiplied by the number of historical points and the number of model parameters and initial smoothing states that are multiplied by 2.

Parameters

Detected Seasonal period and estimates for other parameters that are used in the selected exponential smoothing model are available.

Seasonal period
The number of time steps in a seasonal period used in the exponential smoothing model.

Alpha
The smoothing factor for level states in the exponential smoothing model. Small values of alpha increase the amount of smoothing, that is, more history is considered when the alpha is small. Large values of alpha reduce the amount of smoothing, which means that more weight is placed on the more recent observations. When the alpha is 1, all the weight is placed on the current observation.

Beta
The smoothing factor for trend states in the exponential smoothing model. This parameter behaves similar to alpha, but is for trend instead of level states.

Gamma
The smoothing factor for seasonality states in the exponential smoothing model. Serves the similar role as alpha, but for the seasonal component of the model.

Phi
The damping coefficient in the exponential smoothing model. Long forecasts can lead to unrealistic results, and it is useful to have a damping factor to dampen the trend over time and produce more conservative forecasts.

Diagnostics

Information includes Missing count, Series length, Ignored periods, Trend strength, Seasonality strength, and Date/time interval.

Missing count
Indicates the number of data rows that have either missing values or missing time points and are positioned between the first and the last valid series value. Invalid time points, as well as points with missing values at the first or the last historical time points are not included.

Series length
Indicates the number of data points used for time series modeling. Only the points between the first and the last valid series value are included.

Ignored periods
An integer, m, that ignores the last m data points of the series when building the exponential smoothing model and computing the forecasts. Any missing values at the end of a non-ignored portion of a series will also be forecast. The default value for this parameter is 0, which means that all of the historical data is used in model generation when there are no missing values. A maximum potential of 100 points can be ignored. Ignored periods excludes the data points when building a model, so forecasting might fail due to factors such as minimum data length requirements and missing value proportion that exceeds 33%.

Trend strength
Compare the original model, M, with the same model, but with the trend component removed. The trend strength of M is the difference in accuracy between model M and model M with the trend component removed.

Seasonality strength
Compare the original model, M, and the same model with the seasonal component removed. The seasonality strength of M is the difference in accuracy between model M and model M with the seasonal component removed.

Date / time interval
The Date / time interval represents the detected time interval of the chronologically sorted data. The time interval is identified as the smallest difference between neighboring points in the data when sorted in the chronological order.

Forecasting models

Exponential smoothing models are a popular class of time series models.

Exponential smoothing models are applicable to a single set of values that are recorded over equal time increments only. However, they support data properties that are frequently found in business applications such as trend, seasonality, and time dependence. All specified model features are estimated based on available observed data. An estimated model can then be used to forecast future values and provide upper and lower confidence bounds for the forecast values.

Each model type is suited for modeling a different combination of properties that are found in the data. The model type that can provide the best match to the observed data is selected for modeling the observed data and is used to forecast any future values.

Model estimation algorithms

Models are specified by the smoothing equations that include the model parameters and initial smoothing states. Model parameters are estimated with values that minimize the model error.

Smoothing equations

Exponential smoothing models derive their name from the smoothing equations that specify the model. They provide formulas for computing smoothing states for each observed point by using the current observed value and the previous smoothing states. Smoothing equations provide weighted averages of the current value and the previous states in the time series. Weight for the current value or state is given by a model parameter between 0 and 1, while the weights for previous values are exponentially decreasing.

Level smoothing equations

All model types compute a level state for each time series point by using the corresponding level smoothing equation. Level states for the model without trend and seasonal components are computed as the weighted average of the time series value at the current point and the level state at the previous point. The weight that is associated with the current value is a parameter, **`alpha`**, with its value restricted between 0 and 1. For other models, previous trend and seasonal states are also included in the level smoothing equation.

Trend smoothing equations

Model types with additive or damped additive trend compute a trend state for each time series point by using the corresponding trend smoothing equation. The trend state for the current point is based on the difference of level states at the current and the previous point, and on the trend state at the previous point. The weight that is associated with the difference of level states at the current and previous point is a parameter that is named **beta** with its value restricted between 0 and 1. An extra parameter, **phi**, is added to the damped trend smoothing equations. **Phi** multiplies the trend state contribution from the preceding point and its value is also restricted between 0 and 1. The purpose of this parameter is to estimate the degree of trend damping from one point to the next.

Seasonal smoothing equations

Model types that support additive or multiplicative seasonality compute a seasonal state for each time series point. The seasonal states are computed by using seasonal smoothing equations. The seasonal state for the current point includes the difference of the time series value and the current level state for additive seasonality or ratio of the two same values for multiplicative seasonality. The weight that is associated with this term is a parameter, **gamma**, with its value restricted between 0 and 1. The rest of the contribution comes from the corresponding seasonal state in the previous seasonal period. Notice that the seasonal period has a fixed length, and while the seasonal state can change for each point, only matching seasonal indices from different periods are considered together in the seasonal smoothing equations.

Initial smoothing states

Values must be specified for level, trend, and seasonality states for points that precede the time series. The values are needed for the smoothing equations. To compute the various states at the first point of the time series requires state values at the corresponding previous points.

Model parameters

Each smoothing equation uses corresponding model parameters:

alpha
Controls the level states.

beta
Controls the trend states.

gamma
Controls seasonal indices across seasonal periods.

phi
An extra parameter that is used for specifying the damped trend.

All four parameters have values between 0 and 1. Higher values of **alpha**, **beta**, and **gamma** mean that more recent observations have higher weight, while lower values mean higher weights for older observations. A higher value of **phi** corresponds to a higher degree of dampening the forecast trend.

Model estimation

Model parameters in the smoothing equations are estimated based on the time series data. Parameters cannot be estimated directly by using a formula. They are estimated by an iterative process that searches for parameter values that minimize the model error. The model error is computed as Mean Absolute Scaled Error (MASE). The iterations stop when no further reduction in the model error can be achieved. Corresponding parameter values together with the initial smoothing states fully specify the estimated model. They are used to compute the model states for all other data points and generate the model forecasts by using a corresponding forecast equation.

Forecasting algorithms

A number of algorithms are used in forecasting.

One-step ahead

Every model supports one-step ahead forecasts based on the corresponding forecast equation. One-step ahead forecasts are needed to compute model errors during the model estimation process.

One-step ahead forecasts are computed sequentially for each data point by using computed level and trend states for the current point, and seasonal states for the last seasonal period.

Forecast error is computed by subtracting forecast value at the previous point from the observed value at the current point. Overall model error, which is used for estimating the model, is computed as an average value of absolute forecast errors. Smaller errors correspond to a better model fit. Accuracy measures displayed in **Statistical details** provide several model summaries of the one-step ahead forecast errors.

k-step ahead

k-step ahead forecasts are used to make predictions for any number of future values following the observed time series data. They are based on the same forecast equations as the one-step ahead forecasts for the specified model.

The number of forecast values that are generated is 20% of the length of historical data series by default. You can specify an exact number of values to be forecast in the **Forecast** dialog box. Any missing values at the end of a particular series will also be forecast, but they will not count towards the specified number of forecast periods.

Confidence bounds

Confidence bounds provide the level of uncertainty that is associated with each forecast value. The bounds typically become wider further into the future, because more distant forecasts are less reliable. Confidence bounds provide relevant insights into the future behavior of the observed time series.

Computation of confidence bounds is based on the overall variance of forecast errors that are estimated on the observed data and a factor that depends on the specified model and on the number of steps from the last observed point.

Automated model selection in forecasts

Multiple model types are used to create candidate models for each time series in a forecast. All nine available model types are normally used, except when a seasonal component is absent. There are only three model types available that do not account for seasonality in the data.

The default value, **Auto**, for the Seasonal period option detects seasonal period by comparing multiple models, each with a different candidate seasonal period.

Multiple models are compared by using a model error and the number of model parameters. For example, when model errors are equal for two models, the model with fewer parameters is preferred. The latter model provides a more condensed representation of the observed data and also tends to generate more reliable forecasts.

Forecasting tutorial

This tutorial consists of interactive tasks that help you learn how to use IBM Planning Analytics Workspace to forecast on your data.

This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

Forecasting tutorial step 1: preparing

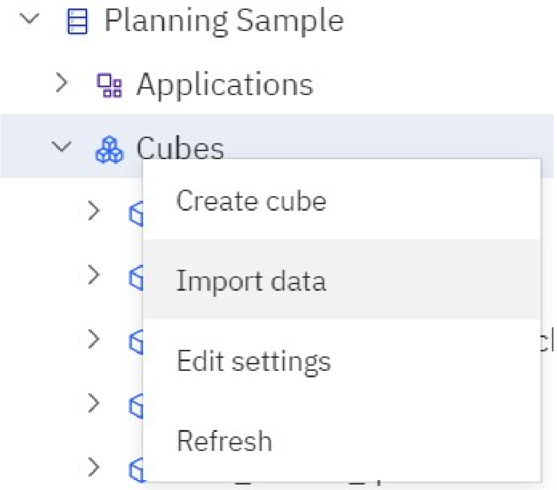
For the forecast tutorial, you download a sample file and set up the sample data in IBM Planning Analytics Workspace.

Before you begin

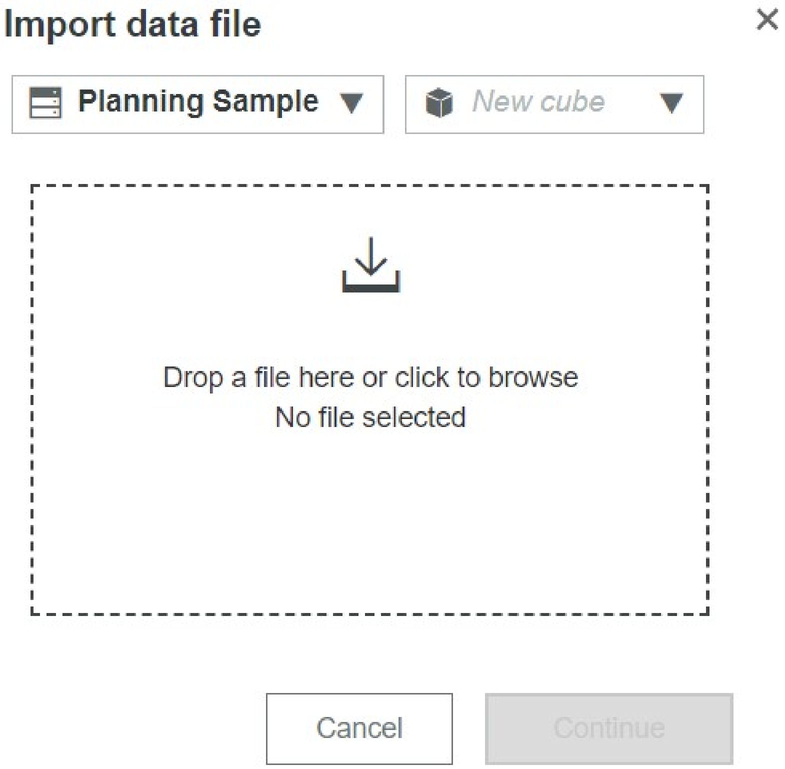
This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

Procedure

1. Download the sample file `forecasting_tutorial.csv` from https://www.ibm.com/support/pages/node/6334299.
2. Log in as an administrator.
3. Create a Planning Analytics Workspace book and expand Planning Sample.
4. Right-click **Cubes** > **Import data**.



5. On the **Import data file** dialog box, select **New cube** from the menu.



6. 5. Select `forecasting_tutorial.csv` and click continue.
7. On the next screen of the **Import data file** dialog box,

a) Enter a name for your cube that you can easily identify. Use your initials as a suffix: `<cube name>_<your initials>`.

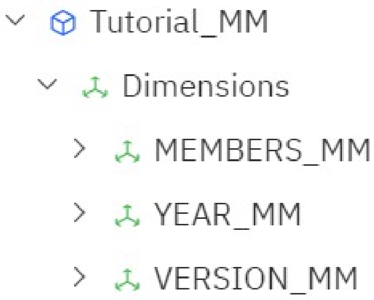
b) Enter the mappings the following way:

By default the screen shows two slots for cube dimensions. Click **Add dimension** to add the remaining one.

Cube dimension	Type	Parent	Child
`MEMBERS_<your initials>`	Leaf only		Members_to_Forecast
`YEARS_<your initials>`	Leaf only		Year
`VERSION_<your initials>`	Leaf only		Version
Data Numeric			Value

c) Click **Overwrite existing data**.

d) Click **Create cube**.



Tutorial_MM

Dimensions

MEMBERS_MM

YEAR_MM

VERSION_MM

The sample cube has four members on the **VERSION** dimension:

Member	Purpose
Actuals	Holds the user data.
Forecast	Typically used to store forecast data, that way the actuals remain untouched, also used to evaluate the quality of a forecast vs the real data.
Forecast_high	Similar to forecast but used to store the forecasted data on the higher bound of the confidence level.
Forecast_low	Similar to forecast but used to store the forecasted data on the lower bound of the confidence level.

The data in the actuals goes in the range 1991 - 2019, but the values after 2013 are all 0; those cells are the ones to be forecasted.

Forecasting tutorial step 2: previewing results

In this part of the tutorial, you preview the results of a forecasting process. The preview operation has no side effects on your data. The preview operation runs on sandboxes and does not change your actual data.

Before you begin

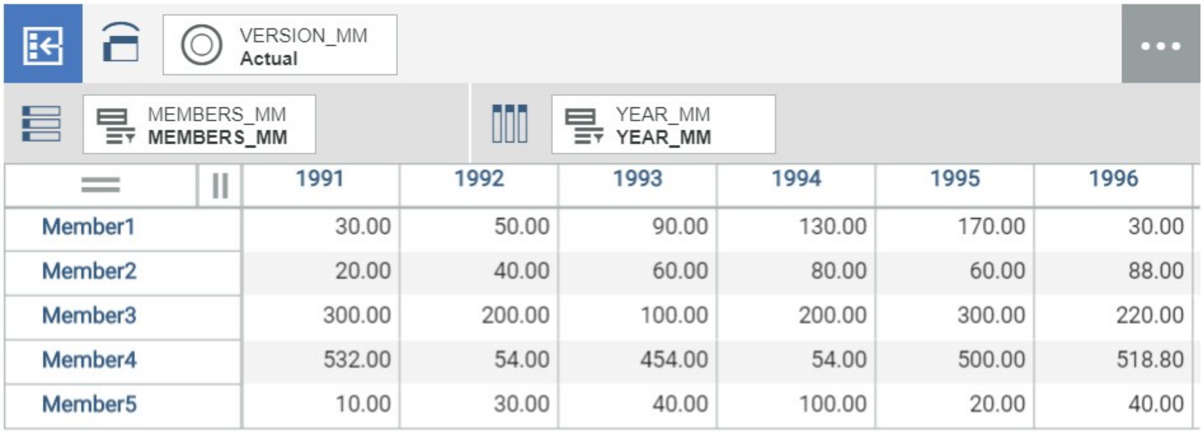
This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

Procedure

1. **Note:** Forecasting requires the time dimension to be placed on the column. The algorithm assumes that the time dimension has no gaps. Forecasting also supports hierarchical time dimensions and nested dimensions as time (that is year, quarter, and month dimensions). Crosstabs views with dimensions that do not have a time dimension on the column can be forecasted, but the process might fail or the results can be unreliable.

   Create a crosstab from the tutorial cube and rearrange until it looks like the following (note that *VERSION* is set to *Actual*):

   - Year on the x-axis.
   - Members that you would like to forecast on the y-axis.
   - Version set to **Actual**.



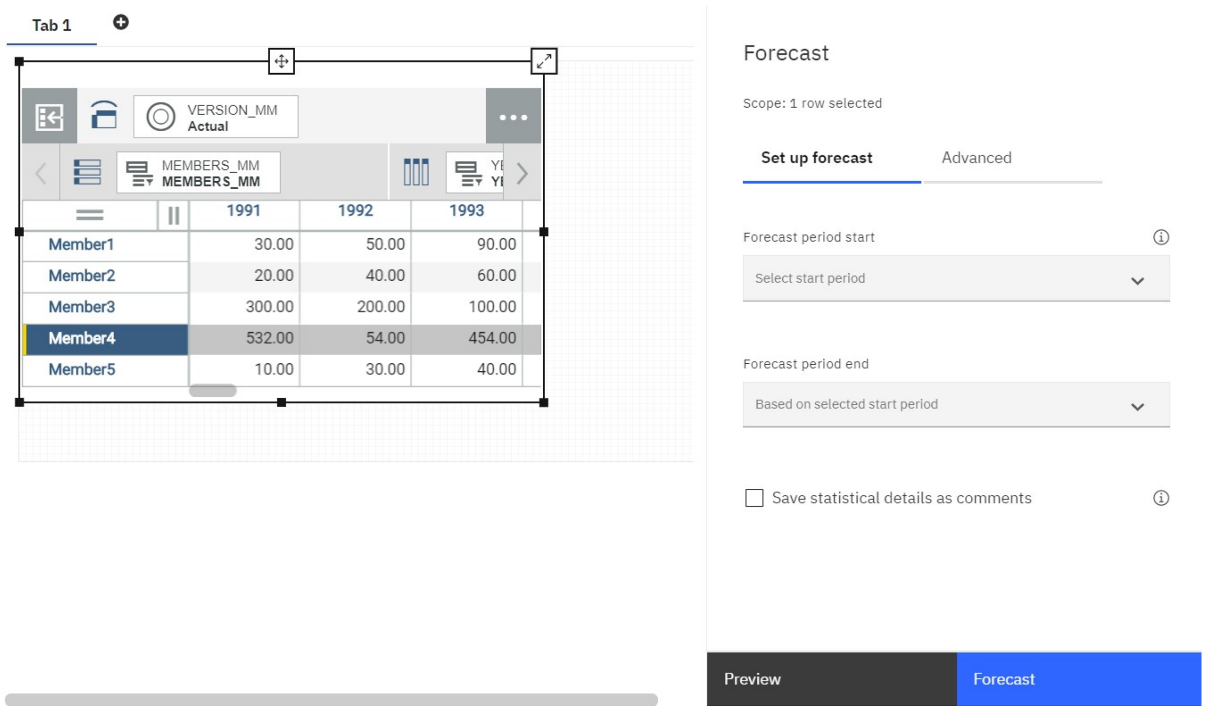
VERSION_MM
Actual

MEMBERS_MM
MEMBERS_MM

YEAR_MM
YEAR_MM

	1991	1992	1993	1994	1995	1996
Member1	30.00	50.00	90.00	130.00	170.00	30.00
Member2	20.00	40.00	60.00	80.00	60.00	88.00
Member3	300.00	200.00	100.00	200.00	300.00	220.00
Member4	532.00	54.00	454.00	54.00	500.00	518.80
Member5	10.00	30.00	40.00	100.00	20.00	40.00

2. Forecasting allows you to preview one time series at the time. Select the view and click **Forecasting** on the toolbar.
3. By default, **Forecast** and **Preview** are disabled. To enable preview click one row. The scope of the forecast is **1 row selected**.



	1991	1992	1993
Member1	30.00	50.00	90.00
Member2	20.00	40.00	60.00
Member3	300.00	200.00	100.00
Member4	532.00	54.00	454.00
Member5	10.00	30.00	40.00

Forecast

Scope: 1 row selected

Set up forecast | Advanced

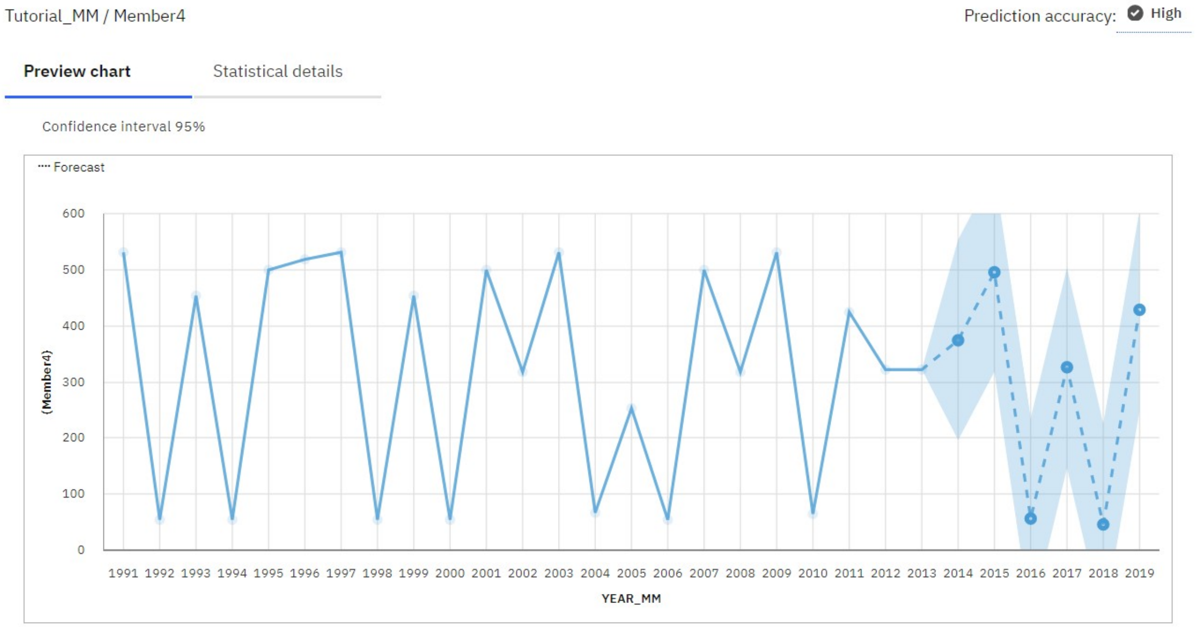
Forecast period start: Select start period

Forecast period end: Based on selected start period

□ Save statistical details as comments

4. Set the **Forecast period start** to 2014. The **Forecast period end** box is populated automatically to reflect the last member on the dimension. In this tutorial 2019.
5. Click **Preview**.

   The forecast preview is displayed.



Tutorial_MM / Member4

Prediction accuracy: High

Preview chart | Statistical details

Confidence interval 95%

6. Click the **Statistical details** tab to display the details of the forecast.

Forecasting tutorial step 3: forecasting based on actuals

In this part of the tutorial, you are going to forecast based on actuals.

Before you begin

This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

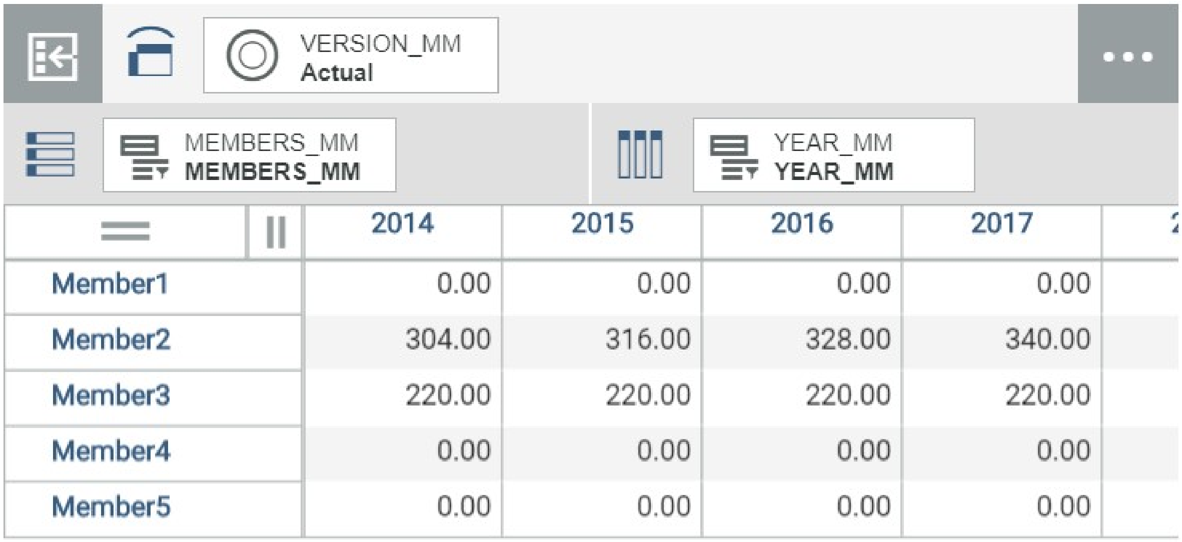
Note: By clicking **Forecast**, you change the TM1 data. The settings in the **Forecast** dialog box are tied to the crosstab used for the process. To share the forecasting parameters with other users, save the book. Different views even the ones that are created from the same TM1 cube do not share parameters.

Procedure

1. Using the same crosstab as in the previous step, select (press Ctrl while you click) 2 or 3 members. **Preview** is disabled, and **Forecast** is enabled. The **Scope** reflects the number of rows selected.
2. Set the **Forecast period start** to 2014.
3. On the **Forecast** dialog box, click **Forecast**.

   It takes some time for the forecast to complete. When completed, a message is displayed that the AI driven forecast completed successfully.

   The data in the crosstab refreshes. Examine the data for the forecasted members. Data is present from the years 2014 - 2019.



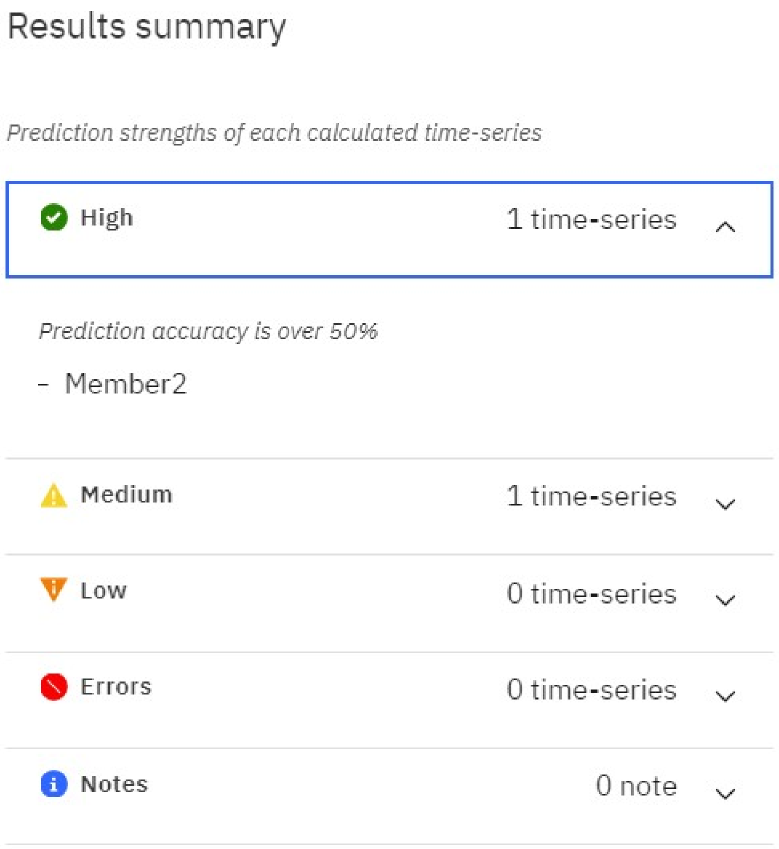
VERSION_MM
Actual

MEMBERS_MM
MEMBERS_MM

YEAR_MM
YEAR_MM

	2014	2015	2016	2017
Member1	0.00	0.00	0.00	0.00
Member2	304.00	316.00	328.00	340.00
Member3	220.00	220.00	220.00	220.00
Member4	0.00	0.00	0.00	0.00
Member5	0.00	0.00	0.00	0.00

The **Result summary** and the **Details** display information on the quality of the forecast. The details specify which time series landed on each quality hierarchy (High, Medium, Low, Errors). The accuracy measure from the forecasting engine defines the hierarchy.



Results summary

Prediction strengths of each calculated time-series

✓ High — 1 time-series

Prediction accuracy is over 50%

- Member2

⚠ Medium — 1 time-series

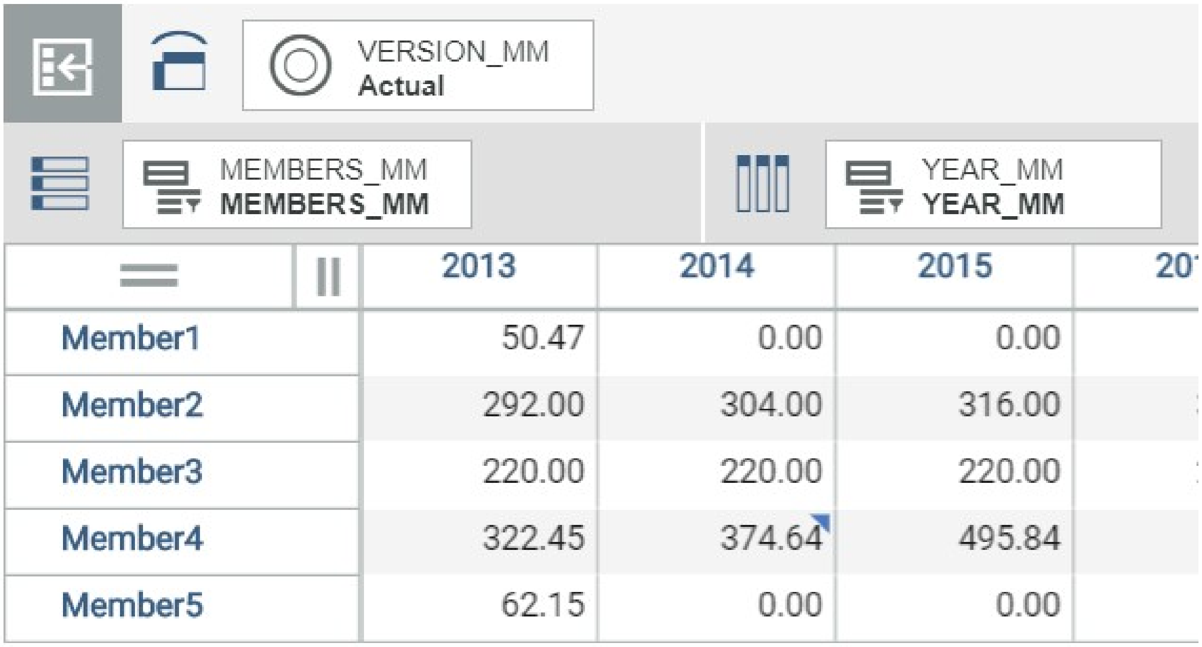
Low — 0 time-series

Errors — 0 time-series

ℹ Notes — 0 note

4. The **Statistical details** tab, shows all the details for the forecast. If you are in the **Details** view on the **forecast** dialog box, click the **Forecast** link, to return to the **Forecast** view.

   You can save the statistical information as a comment. If you do not want the forecasted date to be displayed in cells in the crosstab, use the option **Save statistical data as comments**. For each time-series forecast row, statistical details are saved as a comment on the first forecast cell. Available for up to 25 rows.
5. On the crosstab, select a row or rows that have not been subject of forecasting.
6. On the **Forecast** dialog box, select the option **Save statistical data as comments**.
7. Click **Forecast**.
8. On the crosstab, go to to the first cell that is forecasted on each member. In the example screen capture Member4, 2014. Comments on a cell are indicated by a blue triangular icon on the cell.



VERSION_MM
Actual

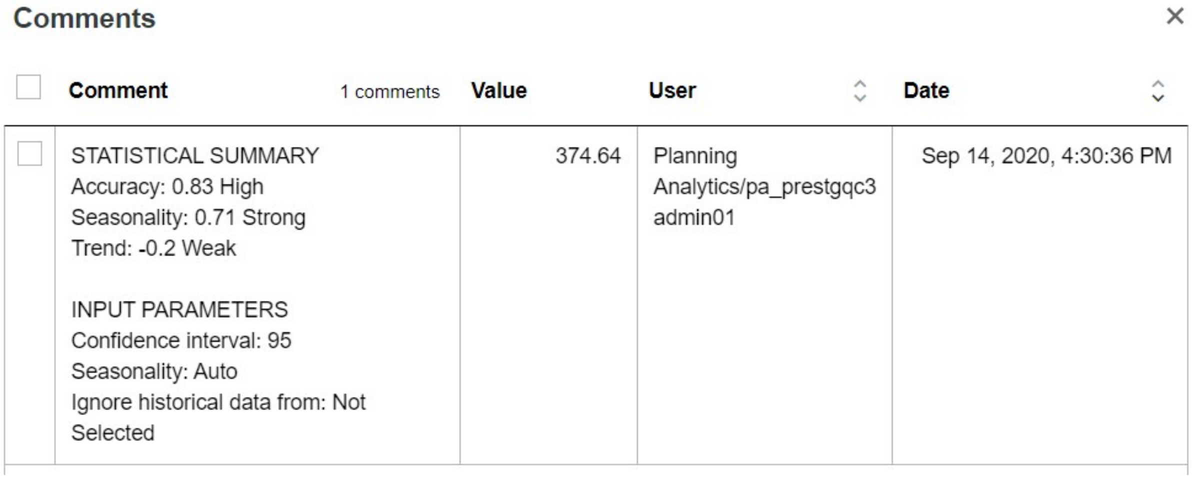
MEMBERS_MM
MEMBERS_MM

YEAR_MM
YEAR_MM

	2013	2014	2015	20
Member1	50.47	0.00	0.00	
Member2	292.00	304.00	316.00	
Member3	220.00	220.00	220.00	
Member4	322.45	374.64	495.84	
Member5	62.15	0.00	0.00	

9. Right-click on the cell and select **Comments**.

   The comments are displayed.



Comments

□	Comment 1 comments	Value	User	Date
□	STATISTICAL SUMMARY Accuracy: 0.83 High Seasonality: 0.71 Strong Trend: -0.2 Weak INPUT PARAMETERS Confidence interval: 95 Seasonality: Auto Ignore historical data from: Not Selected	374.64	Planning Analytics/pa_prestgqc3 admin01	Sep 14, 2020, 4:30:36 PM

10. Close the comments.

    You are going to bring the data back to its state of the start of this topic.
11. On the, all cells for all the members that start on 2014 and ending on 2019.
12. Right-click and select **Clear**.

Forecasting tutorial step 4: forecasting on a different slice

With forecasting, you can select up to three slices to save the output data.

Before you begin

In a typical forecasting process, you might not want to store the forecasting data into the actual slice. With forecasting, you can select up to three slices to save the output data.

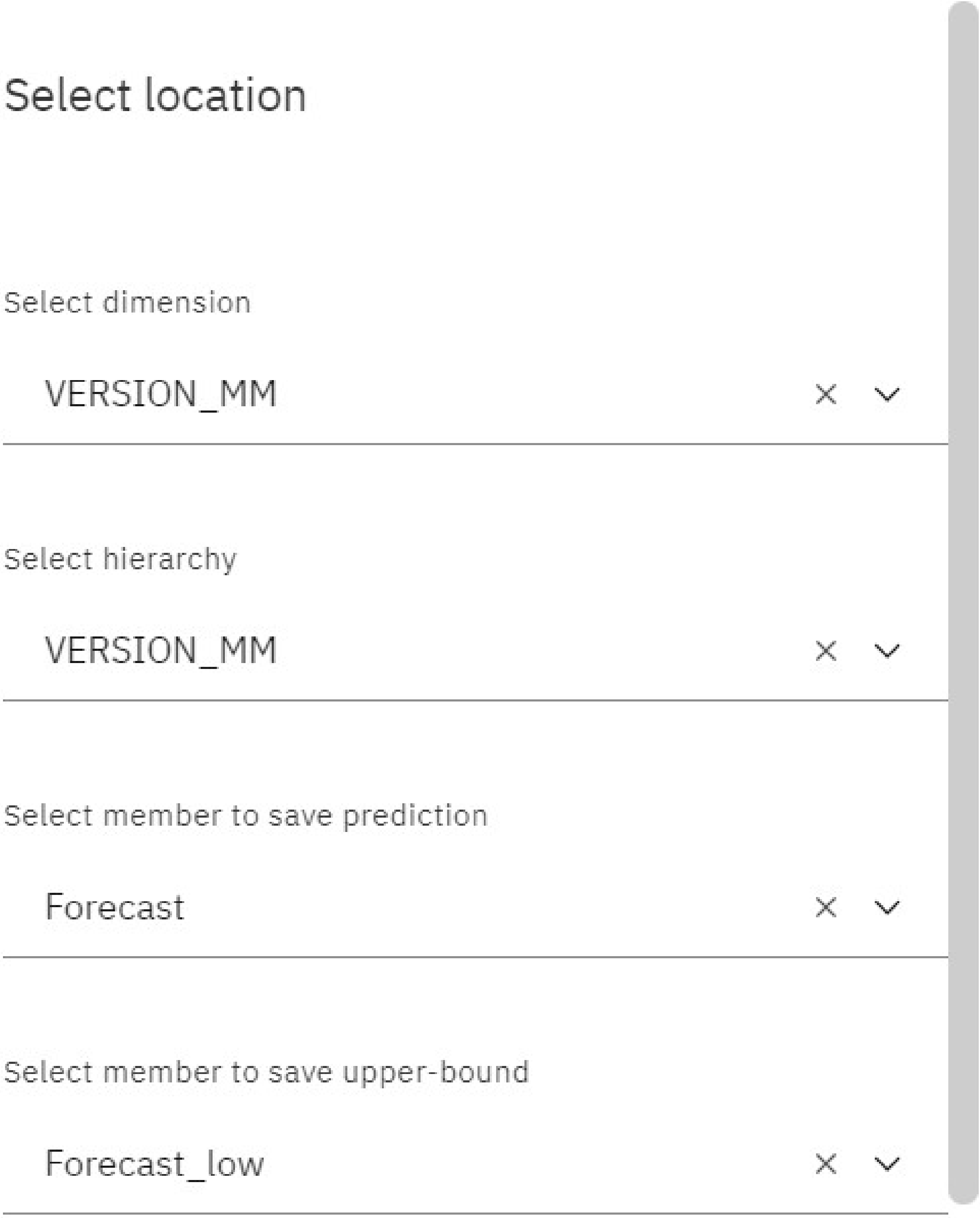
Note: Administrators or modelers must set up forecasting to be saved on a different slice.

This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

Procedure

1. Using the same crosstab as in the previous step, select (press Ctrl while you click) all members. Select only rows, otherwise **Forecast** is not available.
2. Set the **Forecast period start** to 2014. The **Forecast period end** box is populated automatically to reflect the last member on the dimension. In this tutorial 2019.
3. Click the **Advanced** tab.
4. Next to **Where do you want to save the predicted values**, click **Edit**.

   The **Select location** dialog box is displayed.



Select location

Select dimension

VERSION_MM

Select hierarchy

VERSION_MM

Select member to save prediction

Forecast

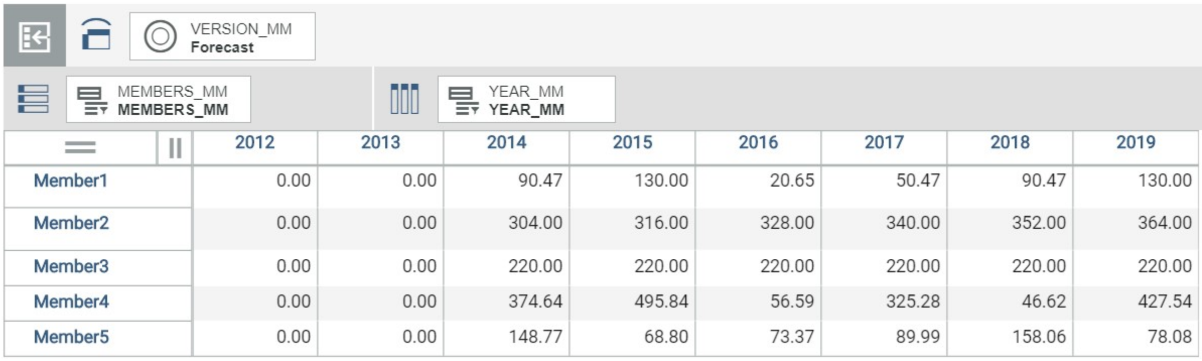
Select member to save upper-bound

Forecast_low

Select dimension	VERSION_*yourinitials*
Select hierarchy	VERSION_*yourinitials*
Select member to save prediction	Forecast
Select member to upper-bound	Forecast-low
Select member to lower-bound	Forecast-high

5. Click **Save**.
6. Click **Forecast**.

   When the forecast is done, look at the data on the crosstab. The values after 2014 remain at 0.
7. In the crosstab, switch the VERSION dimension to the Forecast slice. The data is displayed. The values before the forecast period start -2014- remain in 0 on that slice.



VERSION_MM
Forecast

MEMBERS_MM
MEMBERS_MM

YEAR_MM
YEAR_MM

	2012	2013	2014	2015	2016	2017	2018	2019
Member1	0.00	0.00	90.47	130.00	20.65	50.47	90.47	130.00
Member2	0.00	0.00	304.00	316.00	328.00	340.00	352.00	364.00
Member3	0.00	0.00	220.00	220.00	220.00	220.00	220.00	220.00
Member4	0.00	0.00	374.64	495.84	56.59	325.28	46.62	427.54
Member5	0.00	0.00	148.77	68.80	73.37	89.99	158.06	78.08

8. Examine the values on the other slices.

   If the forecasted members are part of a hierarchy, the resulting values are going to be consolidated by TM1. If you want to save the upper and lower bounds of the calculation, you explicitly set the forecast to do it. Selecting **Save upper and lower bound for consolidations** on the **Advanced** tab.
9. Switch the VERSION dimension to the Actual slice.

   When you set different confidence levels on the **Advanced** tab, you see it impacts the forecast_low and forecast_high values.

Forecasting tutorial step 5: forecasting data with a line visualization

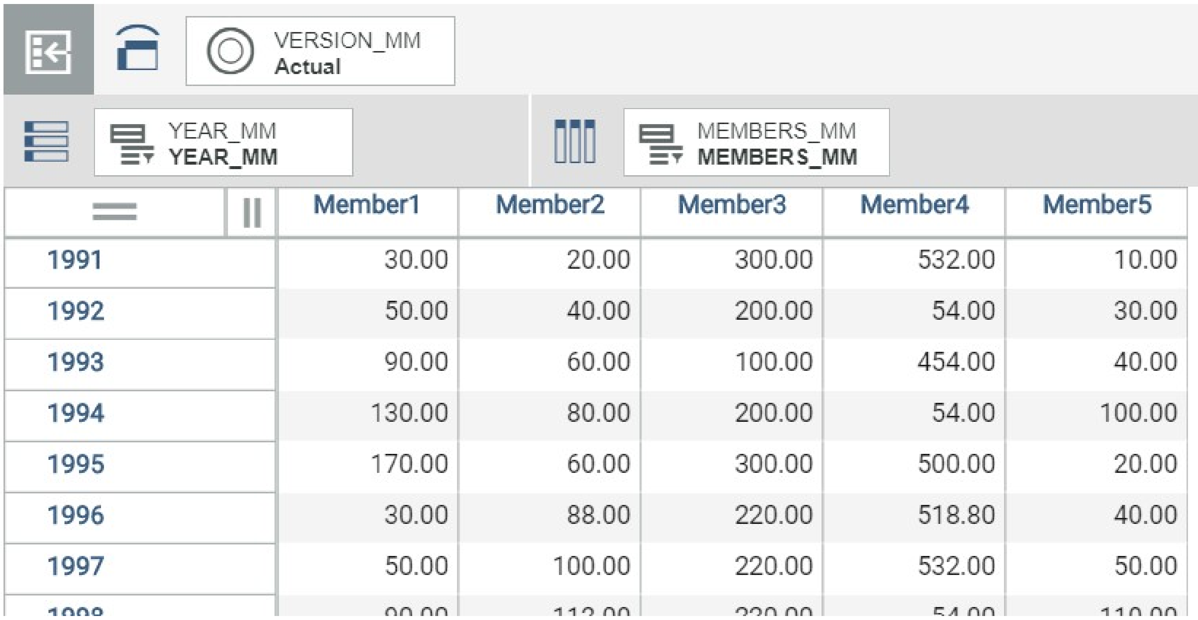
Forecasting includes a line visualization gives a graphical representation of the forecasting results.

Before you begin

This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

Procedure

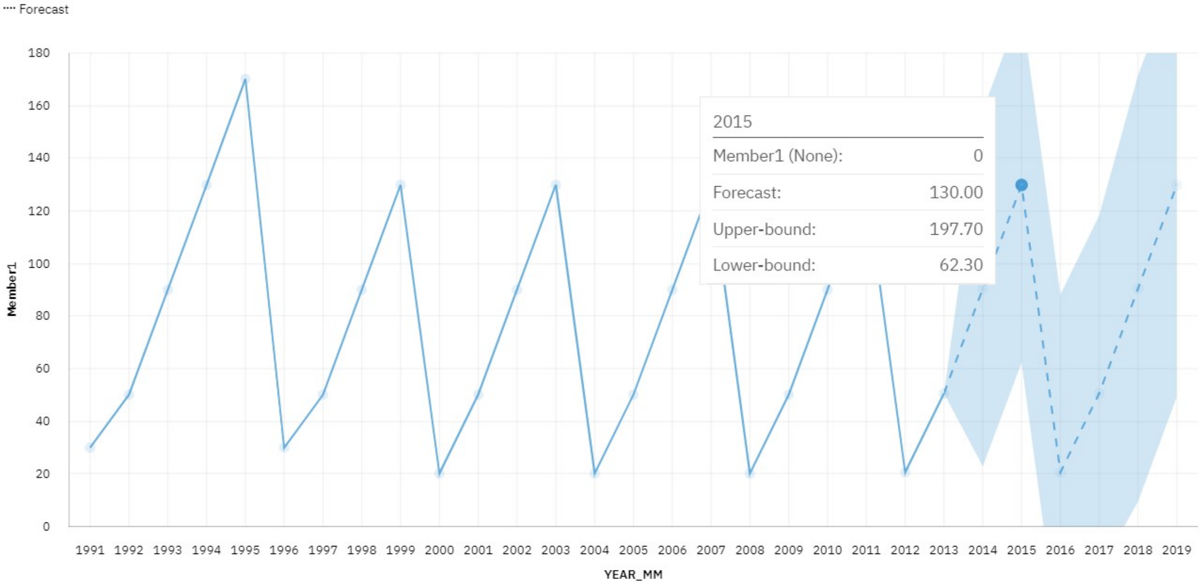
1. In the crosstab, swap the rows and columns. YEAR is on the rows and MEMBERS on the columns.



	Member1	Member2	Member3	Member4	Member5
1991	30.00	20.00	300.00	532.00	10.00
1992	50.00	40.00	200.00	54.00	30.00
1993	90.00	60.00	100.00	454.00	40.00
1994	130.00	80.00	200.00	54.00	100.00
1995	170.00	60.00	300.00	500.00	20.00
1996	30.00	88.00	220.00	518.80	40.00
1997	50.00	100.00	220.00	532.00	50.00
1998	90.00	112.00	220.00	54.00	110.00

2. On the toolbar, click **Exploration** > **All visualizations** > **Line**.

   A line visualization with the forecasting metadata renders. Click any of the forecasted data points to see the details.



2015

Member1 (None): 0

Forecast: 130.00

Upper-bound: 197.70

Lower-bound: 62.30

3. On the toolbar, click **Exploration** > **Exploration**.
4. In the crosstab, swap the rows and columns. MEMBERS are on the rows and YEAR on the columns.

Forecasting tutorial step 6: forecasting on a book set up by another user

Typically, a modeler or administrator sets up the view (crosstab) and the destination slices for contributors to forecast or adjust previous forecasts.

Before you begin

This tutorial is meant to be followed in sequential order. Each next part assumes that the previous part was completed.

You have a book with view on the canvas with a setup forecast pane that includes the forecast destination (advanced tab).

Procedure

1. Save the book as Forecast_tutorial_`yourinitials` in the **Shared** folder. Add the same suffix added to the cube name at the beginning of this tutorial.
2. Sign-out of IBM Planning Analytics Workspace.
3. Sign in as a user with analytics capabilities.
4. From the **Shared** folder, open the book Forecast_tutorial_`yourinitials`.
5. Click the crosstab and then click **Forecasting** .

   The forecasting parameters on the **Setup forecast** and the **Advanced** tab, are present as set up by the administrator on the previous steps.

   Note: The **Edit** link is not available. For that specific crosstab, the forecast is saved on the slices selected by the administrator or modeler who set it up.
6. Similarly, as an analyst you can use the forecast as in the previous exercises. For example, pick a member and click **Preview**.

Synchronize objects in a book or sheet

You can set the level of synchronization that you want between views, visualizations, and other widgets either at sheet level, or at book level. You can set synchronization for dimensions and for sandboxes.

Synchronization is off by default, except for dimension selectors, which are synchronized.

Suppose that you have an exploration (table) view and a stack chart visualization of the same view side-by side on a sheet. You want to focus on car sales in the month of May. With synchronization enabled, when you select May in the Month dimension in the exploration, this selection is reflected in the stack chart. You can disable synchronization of specific dimensions. For example, suppose that you have a Regions dimension. You might want to see Europe in the exploration, but view France in the stack chart, so you would disable the synchronization of the Regions dimension.

Dimension selections, such as drilling down on a member, set editing or selecting, can be synchronized. But actions such as swapping dimensions from row to column cannot be synchronized.

Similarly, if you are using sandboxes to experiment with multiple scenarios, and you want views and websheets to synchronize to a new sandbox selection, you can enable sandbox synchronization.

Procedure

1. In edit mode, select the scope of synchronization. With nothing selected, click and select **Synchronization scope**.
2. Select **Book** to synchronize all sheets in a book, or **Sheet** for objects to be synchronized per sheet.

   Sheet level synchronization is the default.
3. To enable dimension synchronization, select a view, visualization, or websheet, click , and select **Synchronize** > **Synchronize dimensions**.

   Do this for each object that you want to synchronize.

   All dimensions are selected by default. You can clear a dimension if you want to exclude it from the synchronization.

   Tip: Click to close the Synchronize window.

4. To enable sandbox synchronization, select a view, visualization, or websheet, click , and select **Synchronize** > **Synchronize sandbox**.

Data entry

You can enter data by typing in editable cells.

About this task

Read-only cells are colored gray and cannot be updated directly with data entry. Calculated cells are cell values that are derived through rules calculation and cannot be directly edited. By default, cell values derived by rule calculation are colored green, but this depends on the cube view properties set by the modeler.

To enter dates, click in a date cell and select the date.

Consolidated cells are bold. You can spread data by typing a value in a consolidated cell if the **Consolidation TypeIn Spreading**, and **Allow Spreading** capabilities have been set to `Grant` by your administrator. These capabilities are set in the Architect or Perspectives Server Explorer. To find out more, see Capability assignments.

If any descendant leaf cells have a non null-value, a proportional spread is applied.

If all descendant leaf cells are null, an equal leaves spread is applied. An equal leaves spread distributes a specified value equally across the lowest child members of a consolidated cell.

You can use spreading shortcut keys. For example: `S<>100` spreads the value `100` equally to all leaf cells of the consolidated cell, and replaces the existing leaf cell values. You can also use the spreading menu. To find out more, see “Apply data spreading” on page 121.

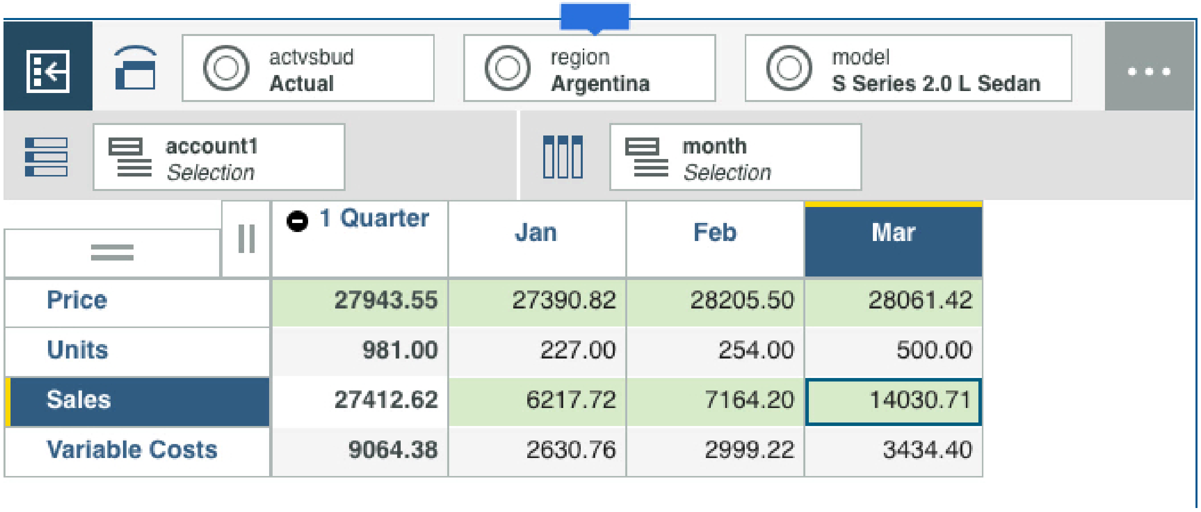
Note: Data in a view is displayed in accordance with your browser locale setting, using the decimal and thousands separators for your locale. You can double-click a cell to view the raw data value as it exists on the Planning Analytics database, which uses a period as the decimal separator and omits a thousands separator. The raw value may include more decimal places than are displayed in the cell.

When you enter data directly in a cell, use the decimal separator for your browser locale.

Procedure

1. In the Price view, notice that the cells in the 1 Quarter column are in bold, so they are consolidated cells. Also, notice that some of the cells are shaded. These shaded cells can't be edited. For example, Sales is a calculated value based on Units and Price, so you can't edit values for Sales.
2. Click the cell at the intersection of Mar and Units. Type `500`, and then click **Enter**.

   Notice that the value of Units under 1 Quarter reflects the change that you made. Notice also that the Sales values for Mar and 1 Quarter have also been updated.



actvsbud Actual

region Argentina

model S Series 2.0 L Sedan

account1 Selection

month Selection

	1 Quarter	Jan	Feb	Mar
Price	27943.55	27390.82	28205.50	28061.42
Units	981.00	227.00	254.00	500.00
Sales	27412.62	6217.72	7164.20	14030.71
Variable Costs	9064.38	2630.76	2999.22	3434.40

3. Click the cell at the intersection of Feb and Units, then type 2K. 2K is a data entry shortcut for the value 2000. You can use all the shortcuts that are described in "Data spread keyboard shortcuts" on page 634.
4. Right-click the cell at the intersection of Jan and Units, then select **Hold**.

   This places a hold on the cell, excluding it from any spreading actions. You can still edit the cell directly. Two vertical bars indicate that a hold is active on a cell ‖ 100.00.
5. Type `S>700` in the cell at the intersection of 1 Quarter and Units.

   To get the same result, right-click in the cell and select **Spread data** > **Equal spread**, enter `700` in the **Value** field, and click **Apply**.

   Notice that 700 is evenly spread by placing the value 350 in the Feb and March children of 1 Quarter, but the value of Jan remains unchanged.
6. Right-click the cell at the intersection of 1 Quarter and Variable costs, then select **Hold**.

   When you place a hold on a consolidated cell, the consolidated value remains constant when you change any of the children. When a consolidation hold is in place and you change the value of a child, proportional spreading is automatically applied to the remaining children so that the consolidated value remains unchanged.
7. Type `2000` in the cell at the intersection of Mar and Variable Costs.

   Notice that 1 Quarter remains unchanged, while Jan and Feb are updated to keep the consolidated value valid.
8. Right-click any cell, then select **Release all holds**. This removes all holds in the view. You can also right-click a cell with an active hold, then select **Release hold** to release a specific hold.

Copy and paste

You can cut, copy, and paste values between views in Planning Analytics Workspace. You can also paste values from other applications, such as Microsoft Excel.

You can use the following keyboard shortcuts to cut, copy, and paste in a view:

Action	Windows	Apple Mac
Cut	CTRL+X	CMD+X
Copy	CTRL+C	CMD+C
Paste	CTRL+V	CMD+V

You can also right-click a cell or a range of cells, and select cut or copy. Paste must be done by using the keyboard shortcut.

When you paste data into a single cell, the paste expands to other cells depending on the number of cells that you copied.

Note: Not all browsers support the right-click menu, but all browsers support the keyboard shortcuts.

Copy a single cell into multiple cells

You can copy a single cell and paste it into multiple cells.

1. Copy the cell.

   Note: To select a range of cells, hold down Shift and select the range of cells.
2. Click CTRL+V or CMD+V to paste.

Copy a range of cells and pasting it into a larger range of cells

You can paste a range of cells into a larger range of cells, repeating the source range where possible. If you copy a range of four cells, the target range must be a multiple of four to repeat the cells. If you copy four cells into a target of nine cells, only the first four cells are pasted into. If you copy four cells into eight writeable target cells, the cells are repeated.

1. For a range of cells, hold down Shift and select the cells. Right-click the selection and select **Copy**.
2. Hold down Shift and select the target cells, then click CTRL+V or CMD+V to paste.

Copy two rows and four columns into two rows and eight columns

This example shows the results of copying two rows and four columns into two rows and eight columns. The header row is not copied.

The source cells that are copied:

1 Quarter	**Jan**	**Feb**	**Mar**
90	25	30	35
87	28	30	32

The result of pasting the source cells into two rows and eight columns:

1 Quarter	**Jan**	**Feb**	**Mar**	**2 Quarter**	**Apr**	**May**	**Jun**
90	25	30	35	90	25	30	35
87	28	30	32	87	28	30	32

Copy rows and columns

You can copy and paste whole rows or columns. You can select a single row or column, or a contiguous range of rows and columns.

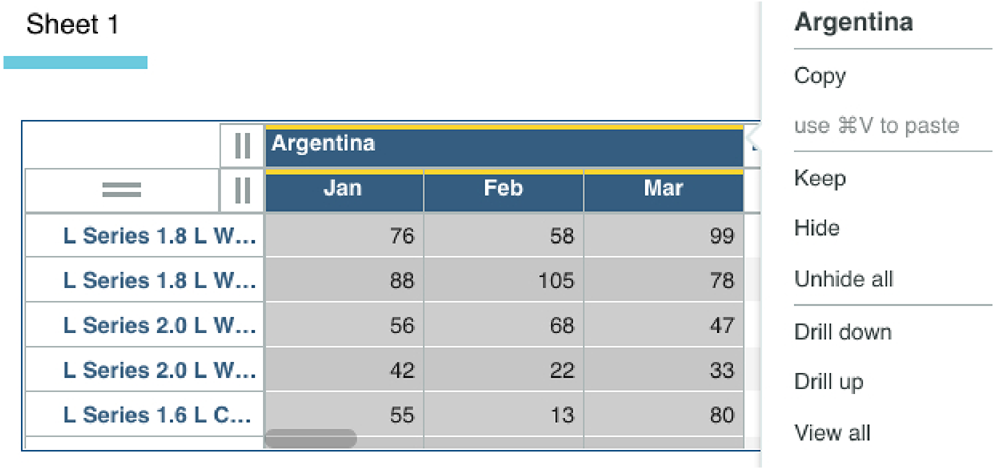
1. Select the row or column headers, then right-click the row or column headers, and select **Copy**.

   It can take a few seconds for data to be copied; a window appears that shows the progress.
2. When the data has been successfully copied, click **OK** in the window, then click the cell where the paste is to start.

   Tip: You can paste the rows or columns into a larger selection, repeating the data, by holding Shift and selecting the target rows and columns, in multiples of the source selection.
3. Click CTRL+V or CMD+V.

Copy and paste from nested hierarchies by selecting rows or columns from the outer axis

You can copy and paste from nested hierarchies by selecting rows or columns from the outer axis. For example, suppose that you have Car models in the rows and Regions and Months nested on the columns. Select Argentina, this selects all the months underneath, and then copy. You can then paste the data into another nested area, such as Belgium.



Sheet 1

	Argentina		
	Jan	Feb	Mar
L Series 1.8 L W...	76	58	99
L Series 1.8 L W...	88	105	78
L Series 2.0 L W...	56	68	47
L Series 2.0 L W...	42	22	33
L Series 1.6 L C...	55	13	80

Copy cells from a column and paste them into rows

In a view without nesting on the rows and columns, you can select a range of cells in a column and paste them into multiple rows. For example, select and copy the values in the Argentina column for the L Series 1.8 L Wagon, L Series 1.8 L Convertible, and L Series 1.8 L Sedan rows. Then, select the cells for Belgium, Brazil, and Chile in the L Series 1.8 L Wagon row, and paste the cells into the selected cells.

	Argentina	Belgium	Brazil	Chile
L Series 1.8 L Wagon	76	0	0	0
L Series 1.8 L Convertible	97	0	0	0
L Series 1.8 L Sedan	56	0	0	0

The result is the selected values are copied into the corresponding cells for Belgium, Brazil, and Chile for the L Series 1.8 L Wagon, L Series 1.8 L Convertible, and L Series 1.8 L Sedan rows, as shown in this table:

	Argentina	Belgium	Brazil	Chile
L Series 1.8 L Wagon	76	76	76	76
L Series 1.8 L Convertible	97	97	97	97
L Series 1.8 L Sedan	56	56	56	56

Copying data from Microsoft Excel

When you copy data from Microsoft Excel and paste it to Planning Analytics Workspace, you must use the same decimal locale separator in both applications. The locales in your browser and Windows must match.

Planning Analytics Workspace is a browser-based application and uses the browser locale setting. Microsoft Excel is a Microsoft Windows application, which uses the language setting in Windows and copies data to the clipboard by using the locale-specific decimal separator.

Copy and paste limitations

There is a 65,536 cell limit for copying and pasting into a view. An error message appears if you try to paste more.

Copy and paste is not supported into or from Websheets in Planning Analytics Workspace.

Paste values to a mixed range of leaves and consolidated cells
You can paste values to leaves and consolidated levels in a view, if the paste special feature is enabled.

The paste special is not available by default. It is available on request only, and replaces the standard paste functionality.

How paste special works

Cells that contain pasted values have the same values as the pasted values; they do not change. The values in leaves that are not pasted change to account for the distribution of the values.

Leaves are pasted first, and each cell is held after the update. Then, cells at the next level up (the first-level consolidation) are pasted to, and they too are held, and so on through the levels.

Comments

You can add comments to cells to add information about the cell value.

You can add a comment to one cell at a time. You can't add comments to calculated cells, or to cells that don't display the original cell value. For example, if the **Show cell value** option is set to something other than **As is**, you can't add a comment.

Procedure

1. Right-click in a cell and select **Comments**.
2. Click **Add comment**, type your comment, and then click **Post**.

   You can add links to web pages in a comment, for example `www.bbc.co.uk`. When the comment is posted, the link is displayed as a hyperlink, so when you click a link, it takes you to the web page.

   After you close the **Comments** window, comments are indicated by a blue triangle in the upper right corner of a cell.
3. To view comments in a cell, right-click the cell and select **Comments**.
4. Sort comments by user (who added the comment), and by date (the default) by clicking the **User** or **Date** heading. To sort on multiple columns, sort on one column, hold Shift, and then sort in the next column.
5. To delete a comment, select the comment in the **Comments** window, and then click **Delete**. You can delete only your own comments. To delete all of your comments, select the check box next to **Comment** and click **Delete**.

Apply data spreading

IBM Planning Analytics Workspace provides a variety of pre-defined data spreading methods that you can use to distribute numeric data to cells in a cube view.

About this task

Predefining data spreading methods can be useful. For example, you can use data spreading to evenly distribute a value across a range of cells or to increment all values in a range of cells by a desired percentage.

When you use data spreading, there are several general steps to follow, which are applicable to all spreading methods.

Step 1 - Select a cell or range from which you want to initiate data spreading: You can initiate spreading from a single cell, a single linear range of cells, or a single rectangular range of cells. You cannot initiate spreading from a non-contiguous range of cells, nor can you spread data across multiple individually selected cells in a view.

Step 2 - Select a data spreading method: After you select the cell or range of cells from which you want to initiate data spreading, you can select a data spreading method. The Spread Data menu is dynamic; it displays only the methods that are valid for the cell or range from which you initiate spreading. For instance, the Relative Proportional Spread, Relative Percent Adjustment, Repeat Leaves, and Equal Spread Leaves methods are not valid for leaf cells. When you initiate spreading from a leaf cell, those data spreading method options appear dimmed on the Spread Data menu, indicating that they are unavailable.

Step 3 - Specify required values: With most spreading methods, you specify only the value you want to spread. However, a few spreading methods require additional values. For example, with the Straight Line method, you specify both a Start Value and an End Value. With the Relative Proportional Spread and Relative Percent Adjustment methods, you must identify a reference cell for the spreading operation.

Step 4 - Select the directions to extend spreading: If you initiate spreading from a single cell, you must specify the direction(s) to extend spreading relative to the point of insertion. The cell from which you initiate data spreading is always included in the spread. Some data spreading methods allow you to extend spreading in multiple directions. These methods display **Direction** options as check boxes, of which you can choose any combination. When you initiate spreading from a selected range of cells, the **Direction** options are disabled and spreading is applied to the selected range.

Step 5- Select an Update action: The **Update action** indicates whether spread values should replace, be added to, or subtracted from existing cell values.

Procedure

1. Right-click the cell or range of cells from which you want to initiate data spreading.
2. Click **Spread data**.
3. Select a data spreading method.
4. Specify the required values and options.
5. Click **Apply**.

   You can also use shortcuts entered directly in cells to apply data spreading.

   See the following links to find out more about each data spreading method.

   - Proportional data spreading
   - Equal data spreading
   - Repeat data spreading
   - Clear data spreading
   - Percent change data spreading
   - Repeat leaves data spreading
   - Equal leaves data spreading
   - Straight line data spreading
   - Growth percent data spreading
   - Relative proportional data spreading
   - Relative percent adjustment data spreading

Hide rows and columns

You can hide rows and columns that are not needed in a view.

Procedure

1. Select the columns or rows that you want to hide, right-click, and select **Hide**.
2. Instead of hiding a member, you can select rows or columns that you want to keep, right-click, and select **Keep** or use the `keep` snap command. All other rows or columns are hidden.
3. To show columns or rows that are hidden, right-click and select **Unhide all**, or use the `unhide` snap command from the toolbar.

   For example, type `hide US, Feb-2004` in the snap command bar. See "Snap commands" on page 160.

Set word wrap options in row and column headers

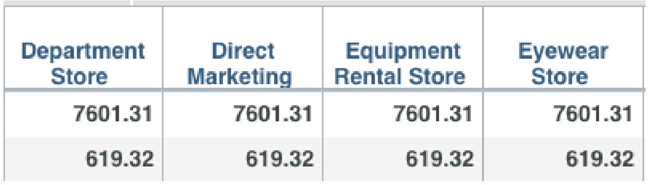
Word wrapping is on by default for column headers and off for row headers. You can also choose to break words, and clear word wrapping.

Procedure

1. Right-click the row selector ═, or the column selector ‖ and select **Word wrapping**.
2. Select one of the options.

Word wrap

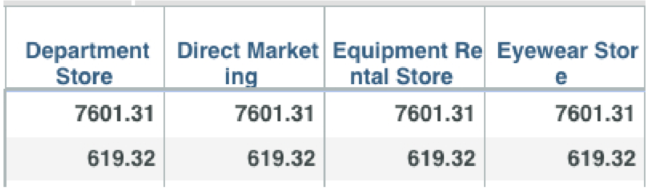
Word wrapping splits a long heading onto multiple lines. The breaks are made at the end of a word.



Department Store	Direct Marketing	Equipment Rental Store	Eyewear Store
7601.31	7601.31	7601.31	7601.31
619.32	619.32	619.32	619.32

Word break

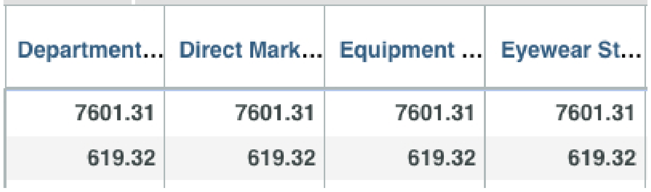
Word breaking also splits a long heading onto multiple lines. Breaks can happen in the middle of a word.



Department Store	Direct Market ing	Equipment Re ntal Store	Eyewear Stor e
7601.31	7601.31	7601.31	7601.31
619.32	619.32	619.32	619.32

Clear wrapping

If word wrapping is cleared, headings that are too long to fit the row or column are truncated.



Department...	Direct Mark...	Equipment ...	Eyewear St...
7601.31	7601.31	7601.31	7601.31
619.32	619.32	619.32	619.32

Make asymmetric selections on rows and columns

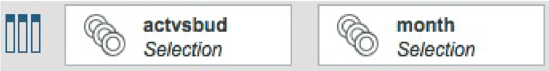
You can make asymmetric selections on rows and columns.

Suppose that you are looking at a view of sales, and you want to see the budget for the fourth quarter. Alongside the budget, you want to see actuals for the first, second, and third quarters. You can do this by making an asymmetric selection.

The following scenario uses the SData database.

Procedure

1. In a new book, go to the SData database, expand **Cubes** > **SalesCube** > **Views**, and drag the **Default** view onto the sheet.
2. Drag the **actvsbud** tile from the Context and drop it in front of the Month tile on Columns so that it is nested:

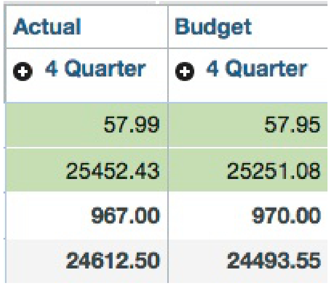


3. Click the month tile and click **Levels** > **level001**.

   You can now see the quarters.
4. Right-click the **Variance** heading, and click **Hide**.

   You can now see the Actual and Budget for the four quarters.
5. Under the Actual heading, click 4 Quarter, then right-click and select **Keep**.

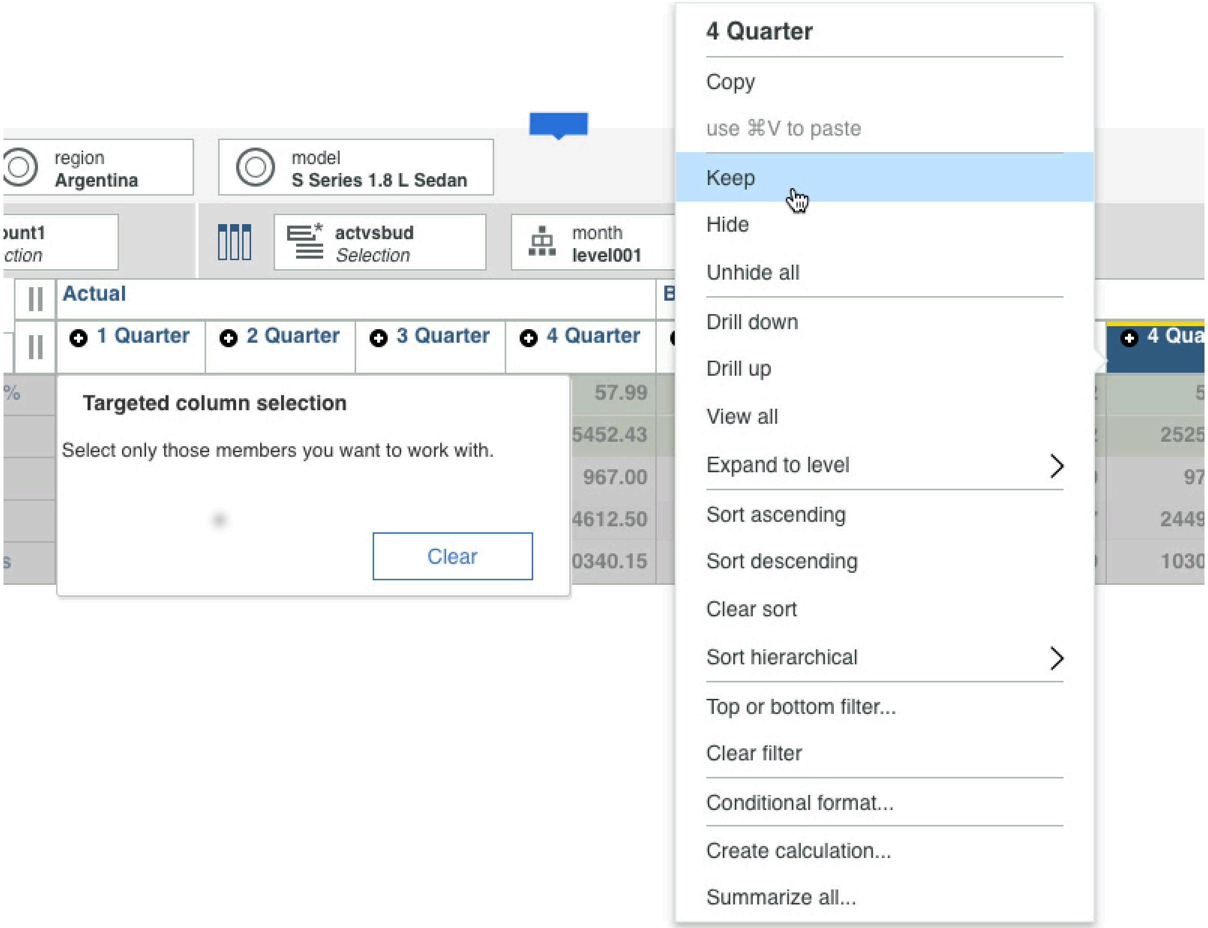
   Notice that the columns for 1 Quarter, 2 Quarter, and 3 Quarter are hidden for both Actual and Budget.



Actual	Budget
⊕ 4 Quarter	⊕ 4 Quarter
57.99	57.95
25452.43	25251.08
967.00	970.00
24612.50	24493.55

   You don't want this, so click **Undo**.

6. To make an asymmetric selection on columns, click the **Targeted column selection** icon.
7. Under the Budget heading, click 4 Quarter, then right-click and select **Keep**.

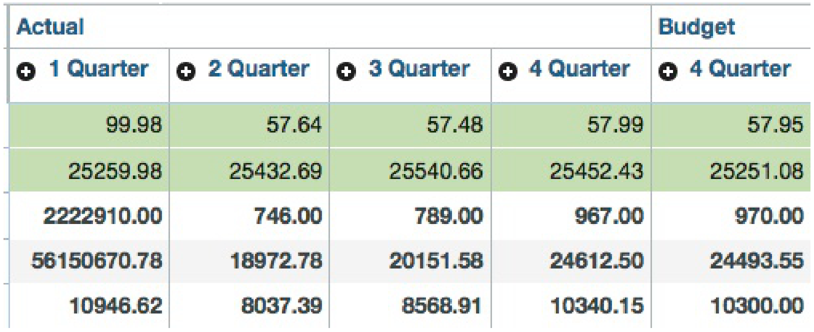


Targeted column selection

Select only those members you want to work with.

Note: To select multiple columns, click **Ctrl** while you click each member that you want to keep or hide in your asymmetric view. You can select only columns or rows that are visible in the view. You can click **Clear** to cancel your selections.

You can now see your budget for 4 Quarter, and your actuals for 1 Quarter, 2 Quarter, 3 Quarter, and 4 Quarter.



Actual				Budget
1 Quarter	2 Quarter	3 Quarter	4 Quarter	4 Quarter
99.98	57.64	57.48	57.99	57.95
25259.98	25432.69	25540.66	25452.43	25251.08
2222910.00	746.00	789.00	967.00	970.00
56150670.78	18972.78	20151.58	24612.50	24493.55
10946.62	8037.39	8568.91	10340.15	10300.00

Drill down

You can drill down on a consolidated member in a view to view the members of the consolidation and to hide other members by double-clicking the member.

You can also drill down on members by using snap commands >_ from the toolbar. For example, type `Drill Georgia` in the snap command bar. See "Snap commands" on page 160.

Procedure

1. To drill up to next level, right-click and select **Drill up**
2. To remove the drills, right-click and select **View all**, or use the `drill reset` snap command.

Drill through to detailed data

You can click a cell in a cube view and drill through to detailed data in another view to get more detail or information about the selected cell. You can then add both views in the same sheet.

The drill-through view can either be a cube view, or an extract from a relational database.

You can drill through multiple levels of cube views, and add the drilled through views to the sheet.

Note: You can only drill to one level when the target is a relational database.

A drill process and drill assignment rule must be defined for a cell before you can drill through to detailed data. For more information, see "Configure drill-through to reveal detailed data" on page 235.

Note: To successfully drill through to an ODBC source, your administrator must create and configure an IBM Secure Gateway.

Procedure

1. To add a drill through view to a book, you must be in edit mode.

   You can preview a drill through view when not in edit mode.
2. Right-click the cell that you want to drill through from, and select **Drill through**.

   The drill through targets are listed.
3. Select the drill through target that you want to see.

   The drill through view opens in a separate view that you can move on the sheet so you can see it next to your original view.
4. To add the view to the sheet, click **Add to current sheet**. Click **Cancel** to close the view.
5. If there are drill through targets to cube views defined in the drilled view, you can drill-though to multiple levels, and you can add one or more views to the sheet.
6. To see which cube and TM1 database the view comes from, hover over this button in the view.

Sort rows and columns

You can sort by dimension member or by data values, in either ascending or descending order. You can choose to sort respecting the hierarchy, or to sort breaking the hierarchy.

You can sort either from the menu, or by using snap commands, see "Snap commands" on page 160.

Procedure

1. To sort by member, right-click either the row selector ═, or the column selector ‖, and select either **Sort ascending** or **Sort descending**.
2. To sort by value, right-click in the row or column and select either **Sort ascending** or **Sort descending**.
3. To sort values within a hierarchy in the dimension, right-click on the hierarchy label, select **Sort hierarchical**, then choose your option.

Show and hide totals

You can show and hide totals in a view, and choose whether totals are leading or trailing.

You can manage totals either from the row or column menu, from the Set Editor, or by using the `totals` snap command. To learn more, see "Snap commands" on page 160.

Procedure

1. Right-click either the row = selector, or the column ‖ selector, and select one of the **Show totals** options.
2. When you edit a set, you can click the **Show Leading/Show Trailing** icon Σ on the Set Editor to toggle the position of totals.

Filter on top or bottom members

You can filter to show only the top or bottom members in a view.

You can see a video that shows how to do this:

https://youtu.be/Bacu0ILVRUI

You might want to show the top three products by territory where the products are different for each territory. You can select how many members to show, and whether to base the sort on the member value, the percentile, or by sum.

Procedure

1. Right-click, or tap and hold the row or column header, and select **Top or bottom filter**.
2. Select whether you want to show top or bottom values.
3. Type the value that you want to filter on. The value depends on whether you are filtering on members, percentages, or the sum.
   - **Members**: Enter a number of top or bottom results. For example, you can type 50 to display the top 50 customers by revenue.
   - **Percent**: Enter a percentage of top or bottom results. For example, you can type 10 to display the customers who contribute to the top 10% of revenue.
   - **Sum**: Enter a number up to which your top or bottom results will sum. For example, you can type 10000000 to display the customers who contribute to the first 10 million dollars of revenue.
4. Select either **Members**, **Percent**, or **Sum**.
5. Click or tap **Apply**.
6. To clear the sort, right-click, or tap and hold the row or column header, and select **Clear filter**.

Suppress zeros

You can hide rows and columns in a view that contain only zeros.

You can suppress zeros in a view from the shortcut bar, by right-click clicking on a row or column selector and selecting **Suppress zeros**, or by using the `zero` snap command.

Procedure

1. To suppress zeros from the shortcut bar:

   a) Click anywhere on the view to display the shortcut bar handle ▬.

b) Click the handle, then click ⊘.

c) Select a zero suppression option.

All
: Suppress all rows and all columns that contain only zeros.

Rows
: Suppress all rows that contain only zeros.

Columns
: Suppress all columns that contain only zeros.

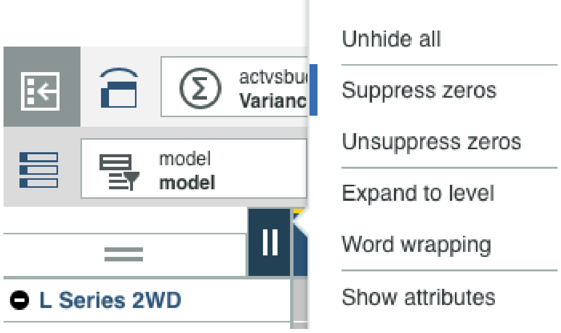
d) Click **OK**.

e) To unsuppress zeros, clear your selection.

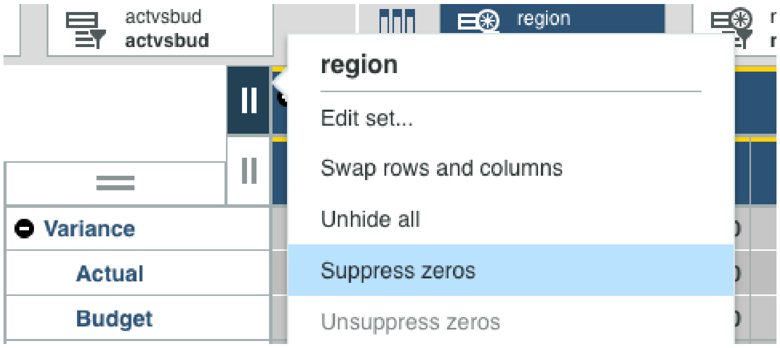
2. To suppress zeros from a row or column selector:

a) Right-click either the row ═ selector or the column ‖ selector, then select **Suppress zeros**.

You can also unsuppress zeros from this menu. A vertical blue line lets you know when **Suppress zeros** or **Unsuppress zeros** is selected.



b) You can suppress zeros on nested rows or columns by right-clicking the selector for the nested row or column and selecting **Suppress zeros**.



Expand levels

You can expand levels in a hierarchy to a specific depth.

You can expand levels manually by clicking ⊞, by selecting a specific level from the menu, or by using the `level` snap command. To learn more, see "Snap commands" on page 160.

Procedure

1. To select a specific level to expand from the menu, right-click the member in the row or column that you want to expand.

   A member that can be expanded has this icon: ⊞.

2. Click **Expand to level**, then select the level.

Display cell values as percentages

You can change the way values are displayed so that values are displayed as percentages of the total.

Procedure

1. Right-click in a cell, and select **Show cell value as**.
2. Select one of the following options:

   % row total

   Displays all the values in each row as a percentage of the total for the row.

   % column total

   Displays all the values in each column as a percentage of the total for the column.

   % grand total

   Displays values as a percentage of the total of all the values or data points in the report.

   Advanced - % parent row total

   Displays the values in each row as a percentage of the parent.

   Advanced - % parent column total

   Displays the values in each column as a percentage of the parent

   The cells that display a percentage are shaded. This shading indicates that the values in these cells are calculated.

3. To return to the actual cell values, in the **Show cell value** window, select **As-is**, or click ↺.

Refresh data

You can refresh data for all views in a sheet or for individual views.

Manually refreshing data

On the dashboard, click ⟳ on the main toolbar, then click **Reload book** to rebuild and refresh all views and websheets across all sheets in the current book.

Note: For versions up to SC2.0.49, click ☰, then click **Reload book** .

To refresh all items on the current sheet, click ⟳ on the main toolbar, then click **Refresh book.**

To refresh an individual view, click anywhere in the view, then click ⟳ **Refresh**.

To refresh the content tree and ensure that you see the most recent views and other objects that are available to you, click ⟳ in the content tree. When you refresh the content tree, the tree collapses to the database level.

Tip: If you frequently work in an area of the content tree, you can bookmark the location so that you can quickly return to it after a refresh. For more information, see "Bookmark items" on page 163.

Automatically refreshing data

You can set options to automatically refresh the data in an individual view. For details, see "Set data refresh options for a view" on page 130.

Set data refresh options for a view

You can set data refresh options directly from the shortcut bar. These options determine which actions in a view trigger a data refresh.

About this task

In versions of Planning Analytics Workspace prior to 2.0.39, data refresh options were set with Grid Refresh view properties. In versions 2.0.39 and later, use the following procedure to set data refresh behavior.

Procedure

1. Click a view, and then click ⏷ to open the shortcut bar.
2. Click the **Refresh data** icon ⟳.
3. Set the data refresh option that you want to apply to the view.

   Select **After each context change** to refresh data after actions that change the structure of the view. This includes actions such as:

   - swapping rows and columns
   - moving hierarchies between rows, columns, context, and the bench
   - expanding or collapsing members in the rows or columns
   - using Snap commands

   If you clear this option, you must manually refresh data after modifying the structure of a view.

   Note: If your view structure changes as the result of a TurboIntegrator process execution, the view is not refreshed automatically when the **After each context change** option is enabled. A view that is modified by a process must be manually refreshed to see the resulting changes.

   Defer on leaf data change controls whether direct edits in leaf cells trigger a data refresh. When the option is selected, data is updated when the view is refreshed (either explicitly or by an action that triggers a refresh), and not when data is changed in leaf cells. Clear this option to refresh view data whenever a leaf cell is modified. This option is off by default when you create a new view in a book.

   Data is always refreshed immediately in a view with any of the following actions:

   - data is entered in consolidated cells
   - data is spread in leaf cells
   - data is spread from the Spread data dialog
   - cells are updated from a date picker

Rollback data entry

You can roll back data entries. Rolling back data entries is useful if you enter data into a cell, which triggers data spreading across a range of cells and you realize that you made the wrong change.

Rollback data entry does not undo any changes that are made to the book, such as swapping rows and columns. To undo book changes, click ↶.

Note: Rollback data entry does not apply to Websheets. Use ↶ to undo Websheet changes.

Procedure

To roll back a data entry, click ⏪.

Rollback undoes the last data entry that you made. To undo previous data entries, keep clicking ⏪. The most recent entries are rolled back first. You can't redo a data entry rollback.

Note:

Sometimes you might not be able to roll back changes. For example, if you commit a sandbox, you can't roll back data entries. Another reason might be that the transaction log for the TM1 database is not enabled. Contact your TM1 administrator for help.

The transaction log is a record of everything that happens on the TM1 database. The log exists up until a **`Save data all`** action is performed on the TM1 database. You can roll back any data entries that are recorded by the transaction log.

Share books and views

You can share books and views, and download books and views as images, PDFs, or Microsoft PowerPoint files. You can also send links.

When you share books or views with people, they can only see the data that they have permission to see, as defined through TM1 security. For example, if a view has a business unit dimension that contains 10 European business units, but a user has rights to see the data from just the UK and Germany business units, only data for the UK and Germany are visible to that user.

When you set permissions for books and views, you can choose the level of access that users can have from one of the following options: **View**, **Edit**, or **Full control**.

When you have the book or view open, you can share it in an email that contains a link to the book or an image of the book. If you save books in folders, you can share the folders.

Note: Email is not available in IBM Planning Analytics Workspace Local.

Procedure

1. Click to share a book or view.
2. Click **Email** ✉ and type an email address, subject, and optionally a message, and then select the format that you want to share. Click **Send** and select one of the following options:
   - **Link**: A link to the book or view is included in the email. The person that you send the link to must have access to the database where the book or view is located.
   - **Image**: An image in PNG format (Portable Network Graphics) is generated and attached to the email.
   - **PDF**: A document in PDF (Portable Document Format) is generated and attached to the email.
3. Click **Download** ⤓ to download the book or view as an image (PNG format), PDF or as a Microsoft PowerPoint. Select the format and click **Download**.

   For a book, you can select which sheets to download.
4. Click **Link** 🔗 to get a link that you can give to another user.

Open Planning Analytics for Microsoft Excel views

You can open and save views created in IBM Planning Analytics for Microsoft Excel if the view is saved to the Planning Analytics Workspace Content Store.

When views are saved to the Planning Analytics Workspace Content Store from Planning Analytics for Microsoft Excel, the view appears as a regular view in Planning Analytics Workspace. Edits and changes can be seen in Planning Analytics for Microsoft Excel after the view is saved in Planning Analytics Workspace.

You open the view from the **Welcome** page. Navigate to the folder where you saved it in Planning Analytics for Microsoft Excel. For example `Personal/My_views`.

Note: Permissions for the view are inherited from the folder that the view is saved in.

Conditional formats

Most financial analysis focuses on highlighting exceptions in rows or columns of data, and not the whole table.

With conditional formatting, you can highlight specific values or cells, by changing what a cell looks like based on a set of conditions that you select. For example, you might want to highlight cells in S Series models that have higher sales than L Series models.

You can apply conditional formatting to both numeric and text values, and you can highlight empty cells by using conditional formatting.

Conditional formatting can also be applied to cell values that have been added to the sheet.

You can watch this video to see a demonstration of conditional formatting.

https://youtu.be/B-0u8j2BHyg

Note: You cannot add conditional formatting on the Apple iPad.

Procedure

1. Ensure that you are in edit mode.

   Note: You can add conditional formats when you are not in edit mode, but you can't save them.
2. Right-click on a row or column header, or cell value that has been added to the sheet, and select **Conditional format**. In the **Conditional format** window, the member in the header that you selected is displayed. Change this selection by clicking the member name and selecting a new member.
   a) Choose one of the operators that you want to apply.

      For numeric values, you can choose one of the following options:

      <
         Less than

      >
         Greater than

      =
         Equal

      <>
         Not equal to

      >=
         Greater than or equal to

      <=
         Less than or equal to

      For text values, you can select one of the following options:

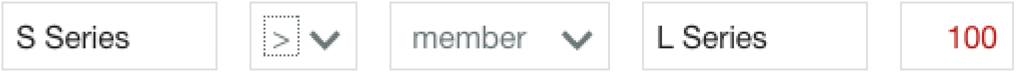
      - Equals
      - Contains
      - Starts with
      - Ends with
      - Is empty

   b) Select whether you want to compare the first member to another member, to a value, or to text.
   c) Select the member, enter the value, or enter the text that you want to compare the first member to. If you want to highlight empty text cells, leave this cell empty.

3. Click ✎ and select what you want the cell to look like. You can select cell format (cell color and border color), font color or style, and icon sets.

   A preview of the formatting is shown.

   For example:



S Series | > | member | L Series | 100

4. Click **Apply**.
5. To add more conditional formats, click ⊕.
6. Reorder the conditional formats by using the arrow keys. ↑ ↓

   The order of conditional formats determines which formatting is used when there are conflicts between formats.

   If you have a format on both a row and on a column, and there are conflicts between the formats, the column formatting applies. For example, suppose that you have a different color background in the row and column, the color selected for the column is used.
7. Click **Save**.
8. To clear formatting, right-click on the column heading and select **Conditional format**. Select the formats that you want to delete, and click 🗑.

Use the keyboard to work with conditional formatting

You can work in the conditional formatting editor by using the keyboard instead of the mouse.

Navigation and selection overview

Move between enabled items

Press Tab to move to the next item.

Press Shift+Tab to go to the previous item.

Open a menu

With the focus on the menu, and press Enter.

Move through items in a menu and members

Press the up and down arrow keys. To move to a sub menu, press the right-facing arrow key.

Select an option

Press Enter.

Enter the formatting area from the conditional formatting editor

The formatting area is where you set the cell format, the text format, and the icon set to be displayed by the conditional format.

Tab to ✎ and press Enter to display the formatting area. Then press Space or Enter to enter the formatting area.

Move between tabbed screens

Press Tab to move between **Cell Format**, **Text Format**, and **Icon Set**.

Move through options in the formatting area

Use the up and down arrow keys to move from one option to another. For example, to move from **Font Style** to **Font Color**.

Save

On a Windows keyboard, press End to focus on **Save**, then press Enter.

On an Apple Mac keyboard, press the up arrow key until you move out of the formatting area, and then tab until **Save** has the focus. Press Enter.

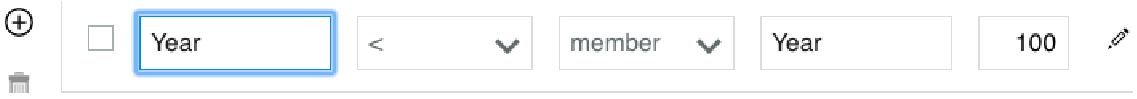
Show or hide formatting area

On a Windows keyboard, when focus is on either **Cell Format**, **Text Format**, or **Icon Set**, press the Home key to focus on ✎ , then press Enter.

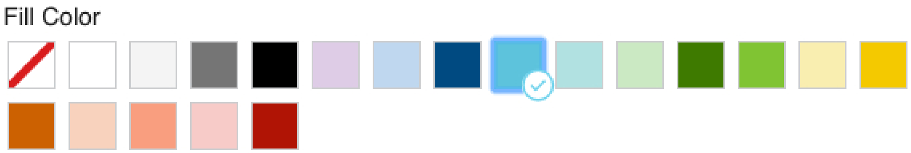
Add a new conditional format by using the keyboard

You can add and edit conditional formats in the conditional formatting editor by using the keyboard.

1. To create the first conditional format, in the editor, tab to the first member tile so that it has focus. The example shows the Year tile with focus.



   Note: To add another new conditional format, tab to **Add**, ⊕ , and press Enter. A new conditional format line is added to the top of the list.
2. To change the member, press Enter to display the list of members. Scroll through the list with the up and down arrow keys, and select a new member by pressing Enter.
3. To change the operator from **<** (less than), tab to the operator selection, press the down arrow key to change the operators, and scroll through the list with the up and down arrow keys. Select a new operator by pressing Enter.
4. To change what you are comparing the first member to from **member** to **value** or **text**, tab to the **member** tile, and press the down arrow key. Scroll through the list with the up and down arrow keys, and select **member**, **value**, or **text** by pressing Enter.
5. If you selected member, to change the member value, tab to the next box and press Enter to display the list of members. Scroll through the list with the up and down arrow keys, and select a new member by pressing Enter.
6. If you selected **value**, or **text**, tab to the empty box and type the appropriate value or text string.
7. To select the condition formatting to be applied, tab to ✎ and press Enter.
   a. Move between the **Cell Format**, **Text Format**, and **Icon Set** tabs by pressing the tab key.
   b. Press Space, or Enter to go into the formatting area. To move between options such as **Fill Color** and **Border Color**, use the up and down arrow keys.
   c. To select a format, such as **Fill Color**, tab to the color. To move to **Border Color**, press the down arrow key until **Border Color** has focus, and tab to the color.



8. To save, on a Windows keyboard, press End to focus on **Save**, then press Enter. On an Apple Mac keyboard, press the up arrow key to move out of the formatting area, then tab to **Save** and press Enter.

Calculations

Add calculations to a view to enhance your analysis. Calculations are saved as part of the view in the Planning Analytics Workspace content store.

You can create two types of calculations:

- **Summary calculations** apply to all visible leaf members and unexpanded consolidated members on a row or column.
- **Member calculations** apply to selected members on a row or column.

This video shows you how to use calculations: https://youtu.be/LCLoGsjr-bg

Add a summary calculation to a view

A summary calculation applies to all leaf members and all unexpanded consolidated members that are visible on a row or column. The calculation updates dynamically as you change the members visible in the view, by drilling down or rolling up consolidations, or by changing the set used in the view.

About this task

You can create summary calculations in Exploration (table) visualizations.

This video shows you how to add a summary calculation to a view

https://youtu.be/jYWMZT7HTqk

Procedure

1. Right-click a row or column label and click **Summarize all**.

   Optionally, you can type a name for the calculation.
2. Click the calculation that you want to create.

   Average
   Displays the average value of the members.

   Minimum
   Displays the smallest value of the members.

   Maximum
   Displays the largest value of the members.

   Median
   Displays the median value of the members (the middle value). If there are an even number of values, it displays the average of the two middle values.

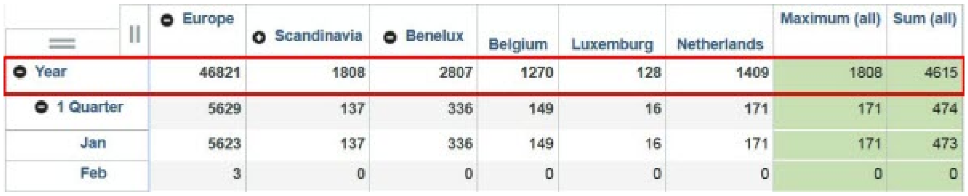
   Sum
   Displays the total of the members.

   Aggregate
   An aggregate calculation allows you to summarize values that can't simply be added up or averaged. For example, percentage or ratio values that are the result of a rule calculation.

   Aggregate displays the aggregate value of the members. This value is affected by whether a member has a C: level rule (a rule that applies to consolidated cells) associated with it or not. If the member has no C: level rule associated with it, Aggregate displays the sum. If the member has a C: level rule, the resulting value is the same as the consolidated member aggregating the same members. To use this feature, you must have IBM Planning Analytics version 2.0.9 or later.
3. Click **OK**.

Maximum and Sum

This example shows the result of applying the **Maximum** and **Sum** calculations to a view that contains a combination of leaf members, expanded consolidated members, and unexpanded consolidated members.



	Europe	Scandinavia	Benelux	Belgium	Luxemburg	Netherlands	Maximum (all)	Sum (all)
Year	46821	1808	2807	1270	128	1409	1808	4615
1 Quarter	5629	137	336	149	16	171	171	474
Jan	5623	137	336	149	16	171	171	473
Feb	3	0	0	0	0	0	0	0

Here, Europe and Benelux are expanded consolidated members. The values for these members are excluded from the summary calculations.

Scandinavia is an unexpanded consolidated member, so its value is included in the summary calculation, as are the leaf members Belgium, Luxemburg, and Netherlands.

The Maximum calculation for Year returns 1808, the value for Scandinavia. Though both 46821 (Europe) and 2807 (Benelux) are greater than 1808, these values are excluded from the calculation because they are expanded consolidations.

Similarly, the Sum calculation for Year returns 4615, which is the sum of Scandinavia + Belgium + Luxemburg + Netherlands. (1808 + 1270 + 128 + 1409 = 4615) Both Europe and Benelux, the expanded consolidated members, are excluded from the calculation.

Member calculations

Member calculations apply to selected members on a row or column.

If a single member is selected, you can choose from the following calculations.

Table 3. Single member calculation options

Calculation name	Description
Rank	Ranks the values of the selected member. Rank 1 is the highest value. Rank is applied to the values visible in the view. If you drill down or roll up consolidations in the view, or if you modify the set used in the view, rankings change to reflect the ordering of the values that are visible in the view. If you attempt to rank a mixture of string and numeric values, the string values are assigned a rank equivalent to a NULL numeric value. String values will always be ranked below any numeric values.
Absolute value	Displays the absolute value for the selected member.
% of total	Displays the member value expressed as a percentage of the root level consolidation (total).
% of parent	Displays the member value expressed as a percentage of its immediate parent.
Member Name +	Enter a numeric value in the entry box. Displays the sum of the member value and the specified numeric value.
Member Name -	Enter a numeric value in the entry box. Displays the difference of the member value minus the specified numeric value. You can click 🔄 to reverse the order of the values, subtracting the member value from the specified numeric value.

Table 3. Single member calculation options (continued)

Calculation name	Description
Member Name /	Enter a numeric value in the entry box. Displays the quotient of the member value divided by the specified numeric value. You can click 🔄 to reverse the order of the values, dividing the specified numeric value by the member value.
Member Name *	Enter a numeric value in the entry box. Displays the product of the member value and the specified numeric value.

If two members are selected, you can choose from these calculations.

Table 4. Two members calculation options

Calculation name	Description
Average	Displays the average of the two member values.
Minimum	Displays the smaller of the two member values.
Maximum	Displays the larger of the two member values.
Median	Displays the median of the two member values.
Aggregate	Aggregate displays the aggregate value of the members. This value is affected by whether a member has a C: level rule (a rule that applies to consolidated cells) associated with it or not. If the member has no C: level rule associated with it, Aggregate displays the sum. If the member has a C: level rule, the resulting value is the same as the consolidated member aggregating the same members. To use this feature, you must have IBM Planning Analytics version 2.0.9 or later.
Member1 * Member2	Displays the product of Member 1 multiplied by Member2.
Member1+ Member2	Displays the sum of Member1 and Member2.
Member1 - Member2	Displays the difference of Member1 minus Member2. You can click 🔄 to reverse the order of the values, subtracting Member1 from Member2.
Member1 / Member2	Displays the quotient of the Member1 divided by Member2. You can click 🔄 to reverse the order of the values, dividing Member2 by Member1.
Member1 % change Member2	Displays the percent change from Member1 to Member2. You can click 🔄 to reverse the order of the values, displaying the percent change from Member2 to Member1.
Member1 % of Member2	Displays the value of Member1 expressed as a percentage of Member2. You can click 🔄 to reverse the order of the values, displaying the value of Member2 as a percentage of Member1.

If three or more members are selected, you can choose from these calculations.

Table 5. Three or more members calculation options

Calculation name	Description
Average	Displays the average all selected member values.
Minimum	Displays the smallest all selected member values.
Maximum	Displays the largest of all selected member values.
Median	Displays the median of all selected member values.
Sum	Displays the sum of all selected member values.
Aggregate	Aggregate displays the aggregate value of the members. This value is affected by whether a member has a C: level rule (a rule that applies to consolidated cells) associated with it or not. If the member has no C: level rule associated with it, Aggregate displays the sum. If the member has a C: level rule, the resulting value is the same as the consolidated member aggregating the same members. To use this feature, you must have IBM Planning Analytics version 2.0.9 or later.

Add a member calculation to a view

Member calculations apply to one or more members on either the row or column axis of a view.

About this task

You can only create a calculation in an Exploration (table) visualization.

The calculations available vary according to the number of members selected. On a mobile device, you can only select a single member for a calculation.

Procedure

1. Select and right-click the row or column label of the members for which you want to create a calculation and click **Create calculation...**.

   Tip: On a mobile device, click the row or column label, then click ⋯.
2. Optionally, enter a name for the calculation.
3. Click the calculation you want to create, see "Member calculations" on page 136.
4. If you are creating an arithmetic calculation on a single member, enter the required numeric value.
5. Click **OK**.

Calculation formats

When a calculation is added to a view, TM1 formatting is applied to the calculation.

The formatting that is applied to a calculation is determined in the following order of priority.

1. The format of the selected members in the calculation. If the calculation uses a range selection, then the format is limited to the start and end member of the range.
2. The format of the overview context members, going from right to left.

   Note: In versions earlier than 2.0.48, the format of the overview context members is determined by going from left to right.
3. The format of members on the bench, going from bottom to top.

   Note: In versions earlier than 2.0.48, the format of members on the bench is determined by going from top to bottom.

4. If no format is found for a member, then a default format of `#,##0.00;(#,##0.00)` is applied. For example, the value 256694.89 is formatted as 256,694.89.

Exceptions

Rank calculations always use the format `#,##0`.

Percentage calculations always use the format `##0.00%;-##0.00%`

Rename a calculation

Though you can assign a name when you create a calculation, you can change the name of an existing calculation.

Before you begin

You must be in the Exploration visualization type to rename a calculation. If you are using any other type of visualization, click the visualization, then click ▬ so that the shortcut bar appears. Then click 📊 and pick **Exploration**.

Procedure

1. Right-click the row or column label of the calculation you want to rename. On a mobile device, click the row or column label, then click ⋯.
2. Click **Rename**.
3. Enter a new name for the calculation.
4. Click **OK**.

Delete a calculation

You can delete one or more calculations from the rows or columns of a view.

Before you begin

You must be in the Exploration visualization type to delete calculations. If you are using any other type of visualization, click the visualization, then click ▬ so that the shortcut bar appears. Then click 📊 and pick **Exploration**.

Procedure

1. Right-click the row or column label of the calculation you want to delete. On a mobile device, click the row or column label, then click ⋯.

   If you are using a desktop browser, you can delete multiple calculations on a row or column simultaneously. CTRL-click or SHIFT-click the labels to select the calculations, then right-click one of the selected labels.
2. Click **Remove**.

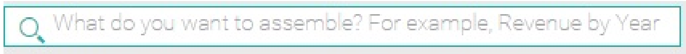
Find data

Find data in IBM Planning Analytics Workspace by using the intent bar or the content tree.

Find and add data using the intent bar

Use the intent bar to find views and data, and add them to a sheet simply by typing phrases or keywords.

When you are working in a book in edit mode, you can find views and data and add them to a sheet by typing phrases or keywords in the intent bar. If you type `view` before your search phrase, only TM1 views are searched. Otherwise all metadata on the TM1 databases is searched.



What do you want to assemble? For example, Revenue by Year

Type `revenue by geography`. If just one match to a view is found, it is added to the sheet. If more than one match is found, or the match is not a view, you can see a preview of all data that matches the phrase and choose what to add.

For example, you can use the intent bar to create a view that contains specific dimensions, as shown in this video:

https://youtu.be/oNzENlOV8-A

Find data with the tree

The tree shows all of the TM1 databases that are configured for Planning Analytics Workspace, and the available cubes, dimensions, hierarchies, sets, levels, views, and members. The tree is hidden by default.

To show the tree, click the show tree icon in the vertical bar: ➕

To hide the tree, click the hide tree icon ➖

You can configure the number of levels that you see in the tree:

Click ⚙ and change the **Navigation depth** to the level you want.

If you have limited the number of levels you see, and you want to see hidden levels in the tree, click the up arrow: ↰

You can make sure that the tree is always synchronized with the view that is currently selected on the screen, by checking the **Always synchronize** option in **Tree settings**.

Sets

Create and edit sets of members in a dimension to limit the number of members that you see in a view. You can also create sets from hierarchies.

This video shows you how to create a simple set.

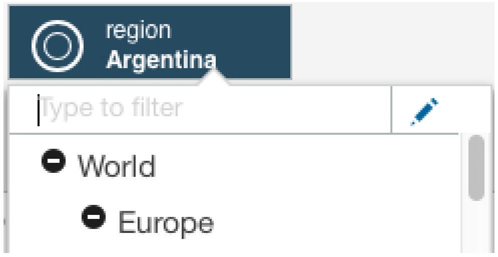
https://youtu.be/aI5cph3E9vk

Edit a set for quick analysis

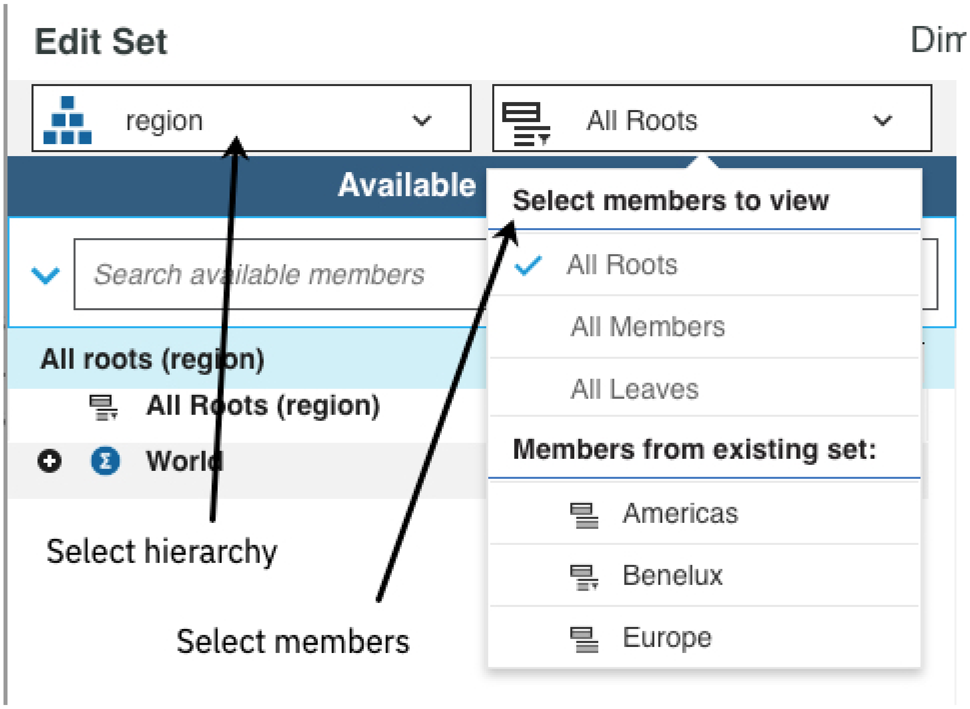
You can edit a set in IBM Planning Analytics Workspace without saving for quick ad-hoc analysis.

Procedure

1. Open the set by clicking the dimension tile in the cube view, and clicking ✎ .



2. Display the members that you want to create the set from by selecting from the **Select hierarchy** menu, and the **Select members to view** menu.



3. If your changes are simple, select the members that you want to be in the set in the **Available Members** pane, and click **Replace and close**. The set editor closes and you return to the view, displaying the members that you selected.

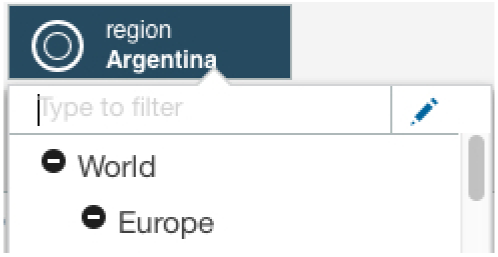
   Note: The **Replace and close** button is not available when you create a new set.

Create and edit sets

A set is a collection of members in a dimension that limits the number of members that are displayed in a view.

Procedure

1. To create a set, right-click a dimension or hierarchy in the content tree, and select **Create set**.
2. To edit an existing set, click the dimension tile in the cube view, and click ✎ .



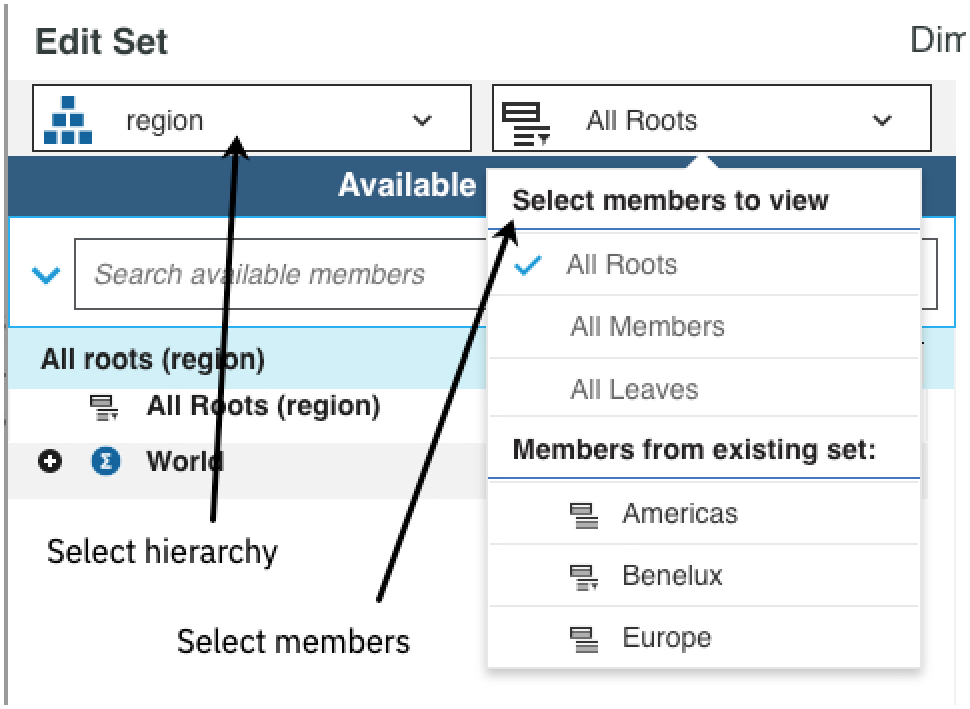
3. You can maximize the set editor by clicking the **Max** icon ⤢.

You can also focus on one area at a time by closing the pane that you are not working on. Click ⊖.

To reopen the pane, click ⊕

Note: If you added the set editor from the Content tree, you can resize the set editor by dragging the grab handles.

4. Display the members that you want to use in the set by selecting from the **Select hierarchy** menu, and the **Select members to view** menu.



Tip: You can choose to display **Default set**, **All roots**, **All members**, or **All leaves** by default. Click ⚙ to access the **Settings** menu.

5. Select the members in the **Available members** list, then choose one of the following options to add the members to the **Current set**:

- Drag the members in the **Available members** list to the **Current Set**.
- Click → to insert the members into the current set.
- Click ⟳ to replace all of the members in the current set.
- To append the members in the **Available members** list to the members in the **Current set**, click ☑, and then click → .
- **Tip:** To control the members that are inserted into the set, click the bullet icon ● and select one of these options.
  - Member only
  - Children
  - Descendants
  - Leaf nodes

- Ancestors

Work in the Current set

6. To change the members in the **Current set**, do the following steps.

   Change the position of a member

   Right-click the member, then click one of the **Move** options. You can only change the position of a member if the set is static. See “Convert a dynamic set to static” on page 149.

   You can also cut ✂, copy 📋, and paste members (use CTRL + V for Windows, or CMD + V for the Mac).

   Remove members

   Select the members, then click ✕.

   To remove all members from the **Current set**, click ☒.

   Keep only selected members

   Select the members, then click ✓.

7. You can display the **Member ID** or the **Alias** for a member instead of the **Caption**. An alias is an alternative name for a member. Click 👤 and select the display name type.

8. You can sort members in the **Current set**.

   Sort ascending

   To sort members in the set in ascending order, click ↑.

   Sort descending

   To sort members in descending alphabetical order, click ↓.

   Sort ascending hierarchical

   To sort members within a hierarchy in ascending alphabetical order, click ↑.

   Sort descending hierarchical

   To sort members within a hierarchy in descending alphabetical order, click ↓.

9. To undo an action in the **Current set**, click ↶. To redo that action, click ↷.

Apply and save your changes.

10. To save your changes as a new set that can be reused in other views, click **Save**, and give the new set a name.

    Select **Share public** to share the set with other users. Clear this option to keep the set private.

11. If you are editing an existing set, you can apply the changes to your view without saving the new set configuration. Click **Apply and close**.

Filter the selection in the set editor

If you have large dimensions, you can make it easier to find what you are looking for in the set editor by filtering the selection in the **Available members** pane.

You can filter the selection in the following ways:

Restrict the content in the set editor to the selected members and their descendants

Select the members in the **Available members** pane, right-click the selection, and select **Keep**.

A selection is created that replaces **All members**. You can search in this selection, and you can use the selection to create a set.

Select one of the default sets

You can choose to display **Default set**, **All roots**, **All members**, or **All leaves** by default. Click ⚙ to access the **Settings** menu.

All roots

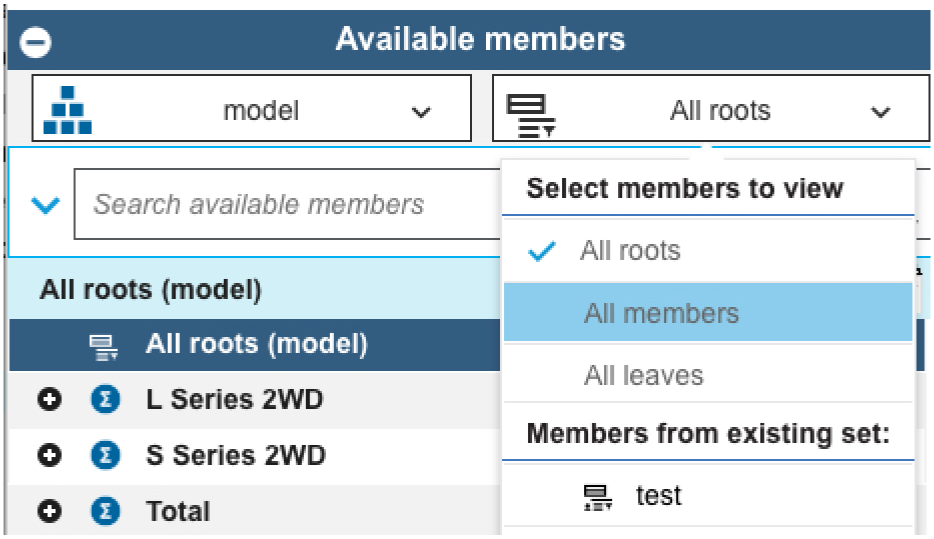
Contains the root members of the hierarchy. The leaf members are hidden but can be displayed by clicking ⊕. When **All roots** is searched, all members, including the leaf members, are searched.

All leaves

Contains the leaf members of the hierarchy. When this set is searched, only the leaf members are searched.

All members

Contains all members in the hierarchy, both consolidation and leaf members.



Drill down on one or more members

Select the members in the **Available members** pane, right-click, and select **Drill down**. This option creates a selection that can be searched in, or used to create a set.

You can also double-click a member to drill down.

Expand selected members and show their children

Select the members, right-click, and select **Expand to descendants**. You can also collapse one or more expanded members by selecting the members, right-clicking and selecting **Collapse**.

Show all ancestors

Select the members, right-click, and select **Show all ancestors**. Shows just the selected member and its ancestors. Other members are hidden. This option creates a selection.

Paste members into the set editor

You can paste both member names and aliases into the set editor, in any combination.

When you paste an alias, the associated member name is inserted into the current set, and the alias is displayed only if aliases are enabled in the set editor.

You can paste only member names that exist in the parent dimension. If you try to paste any names that are not members of the parent dimension, an error message displays a list of those names.

Procedure

1. Copy the member names to the clipboard (CTRL+C for Windows, CMD+C for the Apple Mac) from Microsoft Excel, Planning Analytics for Microsoft Excel, a word processor, or email, or from the **Current Set** area.

In a spreadsheet application, names can be aligned on a single row or column, or on a contiguous rectangular range, but there can be no empty cells within the copied selection. Empty cells cause an error when you paste into the set editor.

In other applications, such as a word processor or email, each member name must appear on a separate line or be on a single line that is separated by tabs. You can also copy member names from a table. However, there cannot be any empty strings (a line without a member, a table cell without a member, a tab location without a member) within the copied selection.

2. Use CTRL+V (Windows) or CMD+V (Apple Mac) to paste the member names into the current set.

   If you paste into an empty current set, the pasted names become the current set.

   If you paste into an existing current set without selecting an existing member as an insertion point, the pasted names are appended to the end of the existing current set.

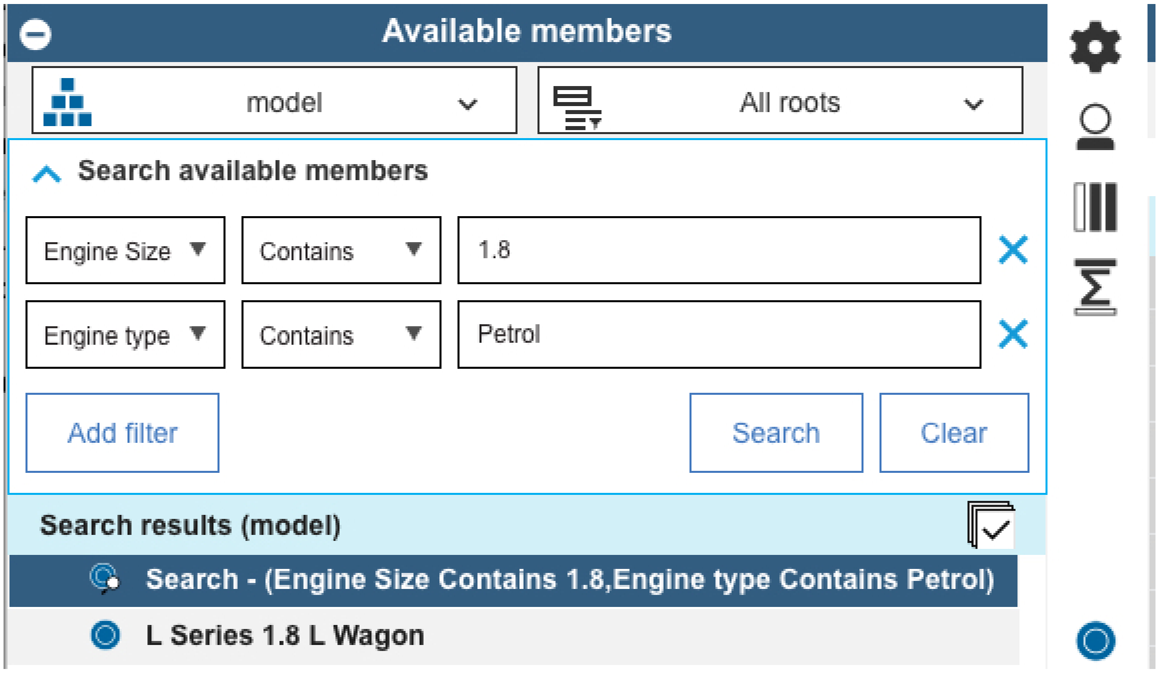
   If you select a member in an existing current set as an insertion point, and then paste into the set:

   - The pasted names are inserted immediately following the selected member, provided the selected member is a regular dimension member (leaf or consolidation).
   - If the selected member is part of a group of members that are returned by a dynamic query (or MDX statement), the pasted names are inserted after the last group member.

Search in sets

You can search in the set editor when the list of members is large, or you are unsure of an exact member name. You can also search to include members in the set that match specific criteria.

You can filter on name, level, or attribute. For example, filter members by one or more attributes so that you can choose what appears in your tables and charts based on their attributes. The following example shows a set that is filtered on two attributes: Engine type and Engine Size.



Engine Size Contains 1.8

Engine type Contains Petrol

Search - (Engine Size Contains 1.8,Engine type Contains Petrol)

L Series 1.8 L Wagon

When you search for members and save a set that includes the search results, a dynamic set is created which contains a query that is run every time that the set is opened. If the parent dimension for the set contains a new member that matches the search, the new member will be included in the set the next time that the set is used.

For example, using search operators (<, >, <=,=>, <>), you can create dynamic selections based on ranges of levels in your organization, accounts, or other hierarchy.

This video shows you how to use some advanced search features of the set editor to edit a set.

https://youtu.be/oYXWZN6nUbk

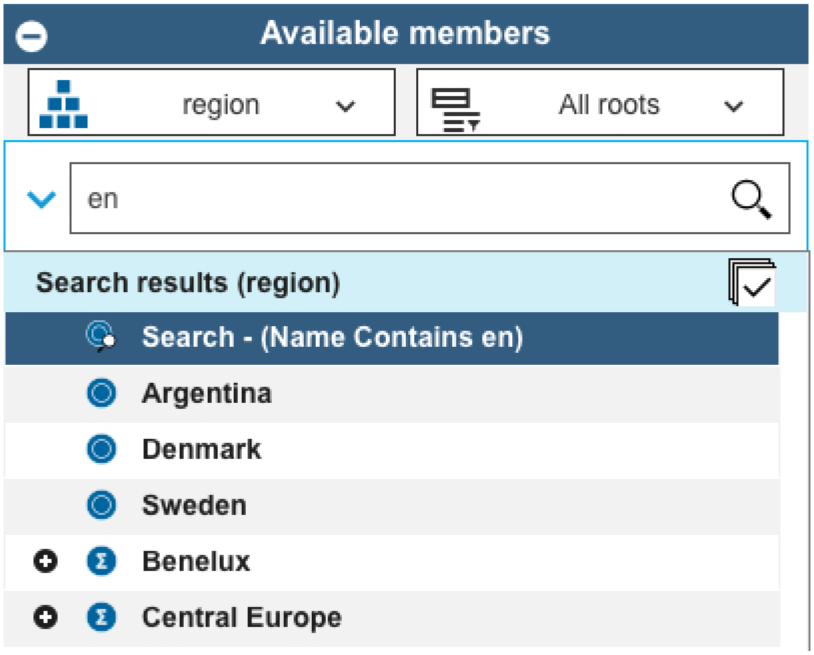
Search for members in a set

You can search in the set editor when the list of members is large, or you are unsure of an exact member name. You can also search to include members in the set that match specific criteria.

Procedure

1. Click the dimension tile and click ✎ to open the set editor.
2. To search for members whose names contain a specific series of characters, type the characters in the **Search available members** box, then click 🔍.

   The Available Members list shows all members that match the search, and a member that is named **Search** that indicates the search criteria. For example, if you search for the characters en in a set of the Region dimension, you get something like this:

   Search - (Name Contains en)
   Argentina
   Denmark
   Sweden
   Benelux
   Central Europe

   When you add the member **Search - (Name Contains en)** to your set, the set includes all current members that contain en. Additionally, the set will include any future dimension members that contain en, such as Greenland or Venezuela.

3. To search on other criteria, click ⌄ .
4. Select the type of criteria you want to search for: **Name**, **Level**, or **Attribute**.
5. Select a search operator (**Contains**, **=**, **<>**, <, <=, >, >=, **Starts with**, or **Ends with**), then type or select the keyword or value you want to search for.

   Tip: If you are searching for a text type attribute and you selected the **=** or **<>** operators, you can select from a list of attribute values from a list. You can also filter the list of values. If you leave the attribute value blank, members with no attribute defined are shown.
6. If you want to add criteria, click **Add filter**, then specify the additional search parameters.

   You can search on up to three distinct criteria.
7. Click **Search**.

   The **Available Members** list shows all current members that satisfy the criteria that you used, and a member named **Search** that identifies the search criteria.

Related topics:

"Convert a dynamic set to static" on page 149

You can convert a dynamic set to a static set. When you convert to a static set, the MDX expression that generated the dynamic set is deleted and the set contains only the members that are present at the time of conversion.

Reorder members in a set
Use the set editor to define the order of the members in a set. Only members in a static set can be reordered.

Procedure

1. Open the set editor for the set that you want to reorder.
2. Click **Members (Selection)**.
3. Click **Convert to static set** ◉.
4. In the **Current set** pane, right-click the member that you want to reorder.
5. Reorder the member in the set by selecting **Move to top**, **Move up**, **Move down**, **Move to bottom**.
6. Click **Apply and close**.

Create a dynamic set
A dynamic set uses Multidimensional Expression (MDX) code. When you create a set from the Content tree, a dynamic set is created by default.
Dynamic sets update automatically based on the evaluation of an MDX expression. The members in a dynamic set can change when new members are added to (or removed from) the parent dimension of the dynamic set.

This feature is for advanced users only.

You can differentiate between dynamic and static sets by looking at the icons in the content tree, dimension tiles in a view, or the set editor: dynamic ≡ and static ≡ sets.

For more information, see MDX Function Support.

This video shows you how to modify a simple MDX expression:

https://youtu.be/9XGC_hxi5lg

Work through this procedure to find car models that constitute the top 20% of sales in a particular region with the SData database.

Procedure

1. Create a book.
2. Navigate to the SData database, SalesCube cube, and expand **Views**. Right-click the All view and select **Add view**.
3. Click the model dimension tile, and then click ✎ .
4. Click **MDX** .
5. Replace the existing MDX expression with this code:

```
TOPPERCENT(TM1FILTERBYLEVEL(DESCENDANTS({[model].[model].[Total]}) , 0),
20.000000 , [SalesCube].([actvsbud].[actvsbud].[Budget],[region].[region].
CURRENTMEMBER,[account1].[account1].[Units],[month].[month].[Year]))
```

6. Click **OK**, and then **Apply and close** to return to the view.
7. The following steps verify that the models that you see constitute at least 20% of sales.

   a) Click the Model dimension tile, and then click ✎ .

   b) Set the insertion option ○ to Member only.

c) Move the Total member in the **Available Members** pane over to the **Current Set** by dragging it across..

d) Right-click the Total member, then click **Move to top**.

e) Click **Apply and close** to return to the view.

f) Right-click and select **Show cell value as** > **% column total**.

The value of the leaf member should be 20% or higher.

Related topics:

"Convert a dynamic set to static" on page 149
You can convert a dynamic set to a static set. When you convert to a static set, the MDX expression that generated the dynamic set is deleted and the set contains only the members that are present at the time of conversion.

"MDX string expressions" on page 148
If you are using MDX expressions to create a dynamic set, you must be aware of the parsing rules for string literals. Special characters and escape characters must be handled carefully in MDX string expressions.

MDX string expressions

If you are using MDX expressions to create a dynamic set, you must be aware of the parsing rules for string literals. Special characters and escape characters must be handled carefully in MDX string expressions.

You must use a syntax notation that works correctly for all cases when special characters are embedded in the strings.

Note: Using single-quoted string literals is preferred. However, Planning Analytics also supports double-quoted string literals. Using double-quoted string literals requires care when you use MDX because the JSON parser that is used for TM1 REST API also uses the double quotation mark as a string delimiter.

Example with single quotation marks in a string

Consider the string literal `O'Reilly`.

- Using single-quoted string literals, `O'Reilly` must be represented as `'O''Reilly'`

  The string must use a single quotation mark to escape the single quotation mark.
- Using double-quoted string literals, `O'Reilly` must be represented as `\"O'Reilly\"`

  The string must use a back-slash to escape the double quotation mark. The single quotation mark in `O'Reilly` does not need escaping.

Example with double quotation marks in a string

Consider the string literal `O"Reilly`.

- Using single-quoted string literals, `O"Reilly` must be represented as `'O\"Reilly'`

  The double quotation mark in `O"Reilly` must be escaped to satisfy the JSON parser.
- Using double-quoted string literals, `O"Reilly` must be represented as `\"O\"\"Reilly\"`

  The double quotation mark in `O"Reilly` must be double escaped; once for the JSON parser, and again for the MDX parser.

As you can see in these examples, using single-quoted string literals is simpler and preferred.

Convert a dynamic set to static

You can convert a dynamic set to a static set. When you convert to a static set, the MDX expression that generated the dynamic set is deleted and the set contains only the members that are present at the time of conversion.

About this task

Converting a dynamic set to a static set is useful when you have added a calculation to a dynamic set, but want to control the positioning of the calculation in the set. When you add a calculation to a dynamic set, the position of the calculation in the set is determined by the MDX expression and the position cannot be changed. When you convert to a static set, you can move the calculation to any location in the set.

Procedure

1. Open the dynamic set in the set editor.
2. Click ◉
3. Click **Save** to save the set as static.

View properties and attributes in a set

You can view properties and attributes for members in the set editor.

You can view any defined attributes:

Weight
The weight property controls how a child member rolls up to its immediate parent, whether that child is also a parent of another consolidation or a leaf member. Top level consolidations do not have a weighting.

Level
Show the level of the member in the hierarchy.

Type
See whether a member is consolidated or simple (leaf).

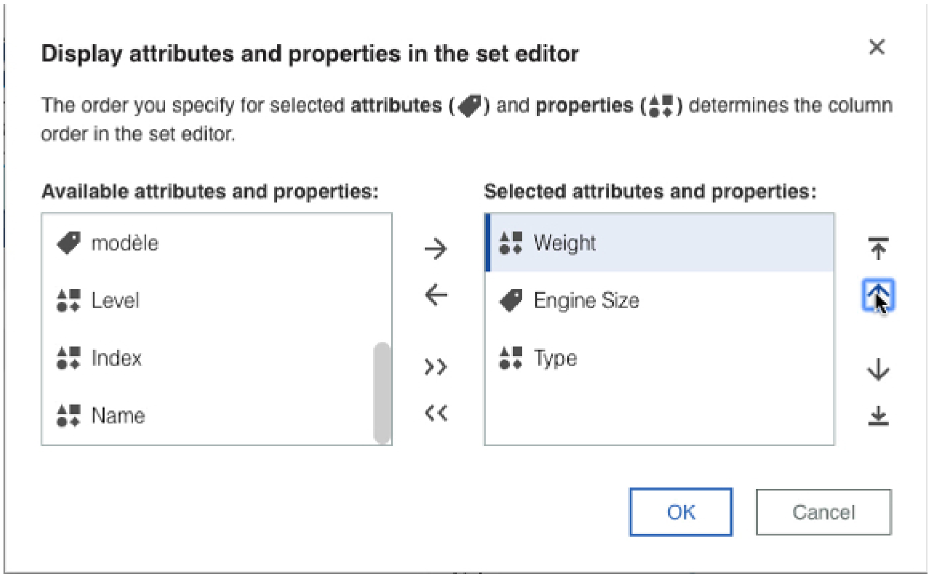
Index
The display order of members in a dimension.

Name
The member ID.

Procedure

1. In the set editor, click ▯▮▮.
2. Select the attributes 🏷 and properties ▲■●◆ in the **Available attributes and properties** pane, and move them across to the **Selected attributes and properties** pane.

   You might need to scroll down to view all properties. The order in which the attributes and properties are listed on the right pane determines the order in which they are displayed in the set editor. You can change the order by using the arrows.



Display attributes and properties in the set editor

The order you specify for selected **attributes** (🏷) and **properties** (▪) determines the column order in the set editor.

×
Available attributes and properties:
modèle
Level
Index
Name
Selected attributes and properties:
Weight
Engine Size
Type
OK
Cancel

Delete a set

You can delete sets that are not currently used in any views. If you are a TM1 administrator, you can delete any public set or any private set that you own. If you are not an administrator, you can delete only private sets that you own.

Procedure

1. In the navigation tree, locate the set you want to delete. The path to a set is ***<Database_Name>*** > **Dimensions** > ***<Dimension_Name>*** > ***<Dimension_Name>*** > **Sets** > ***<Set_Name>***.
2. Right-click the set, then click **Delete Set**.

Results

After you delete a set, the navigation tree contracts to the database level.

Related topics:

Understanding Administrative Groups and Authority

Use the keyboard to work in the set editor

You can work in the set editor with the keyboard instead of the mouse.

Navigation and selection overview

Move between enabled items in the set editor

Press the Tab key to move to the next item.

Press Shift+Tab to go to the previous item.

Open a menu

With the focus on the menu, press Enter or the Space bar.

Move through items in a menu and members

Press the up and down arrow keys. To move to a sub menu, press the right-facing arrow key.

Select an option

Press Enter.

Select consecutive members (multi-select)

Highlight the first member that you want to select, hold Shift, and then use the arrow keys to select the next members.

Expand and collapse consolidated members

Press Enter, or the right arrow to expand consolidated members, and press Enter or the left arrow to collapse.

Show a context menu

On a Windows keyboard, press Shift+F10. On an Apple Mac keyboard, press Shift+Fn+F10. Use the up and down arrow keys to move through the items.

Work in attributes menus

Tab to ▥ and press Enter. Press the up and down arrow keys to move through the menu, and press Enter to select an attribute. Tab to **OK** and press Enter to close the menu.

Navigate in the MDX Editor

To move around the controls, press the Tab key to move to the next control and press Shift+Tab to go back to the previous control.

When the focus is inside the rich text editor, on a Windows keyboard, press CTRL+F2 to move to the next control, and Shift+CTRL+F2 to go to the previous control. On an Apple Mac keyboard, press CMD +fn+F2 to move to the next control, and Shift+fn+CMD+F2 to go to the previous control.

Navigate in the set editor by using the keyboard.

Press Tab to navigate through the items in the set editor. To move backwards through the items, hold Shift and press Tab.

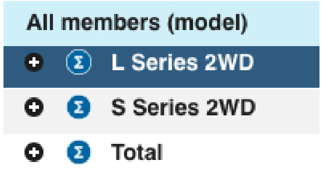
When an area of the screen has focus, it has a contrasting line around it. When you open the set editor, the first highlighted item is the **Available Members** pane.

To open a menu for an item that has focus, press Enter. You can move through the items in the menu by pressing the down and up arrow keys. When a menu item is selected, it has a blue background.

Select members in the Available Members pane by using the keyboard

Tab to the **Available Members** pane. A black line appears around the pane when it has focus.

To select members in the **Available Members** pane, press the down and up arrow keys until the member has focus. A member with focus has a dark blue background.



All members (model)

L Series 2WD

S Series 2WD

Total

Press Enter to expand a consolidated member, and press Enter again to close it.

To select consecutive members, highlight the first member that you want to select, hold Shift, and then use the arrow keys to select the next members.

When you highlight members, they stay selected so you can select either insert → or replace ⟳ to move the members to the **Current Set**.

Search with filters in the set editor by using the keyboard

To create filters to search for members in the **All members** pane with the keyboard, do the following steps:

1. Press Tab until ⌄ has the focus.

   Note: Depending on which item in the Set Editor has focus, it can be quicker to use Shift+Tab to move to ⌄ .

   Then, press Enter to display the search filters.

2. To set a filter, with the focus on **Name**, press Enter, then press the down arrow key to move through the **Level** and **Attributes** menu items. Press Enter to select a menu item.
3. If **Attributes** has a sub menu, press the right arrow key to move the focus to the sub menu. Use the up and down arrow key to move through the items, and press Enter to select an item.
4. After you select either **Name**, **Level**, or **Attribute**, press Tab to move to the next filter. Focus on either **Contains** , **=** or **<>** by using the down or up arrow keys, and press Enter. Then, tab to the next field and type the keyword that you want to search on.
5. Tab to **Add filter**, then press Enter to add a filter and repeat the previous steps.
6. To begin the Search, press Tab to go to **Search**, then press Enter.

Create selectors

You can create selectors for dimensions, members, sets, or levels. For example, suppose that you have a visualization and the overview is hidden. You can add a selector to the sheet so that you change the point of view.

Add a selector by right-clicking the dimension in the tree that you want to use as a selector and selecting **Add as selector widget**.

Procedure

1. In a new book, navigate to the Planning Sample database, and expand **Cubes** > **plan_BudgetPlan** > **Views**, and drag the **Budget Input Detailed** view to the sheet.
2. Click the **plan chart of accounts** tile, and then click ✎ .
3. Under **Available Members**, click ⊞ to expand **Net Operating Income**, then select **Operating Expense**.
4. Click the bullet icon ◎ and select **Children**. Selecting **Children** means that the children of the member are inserted into the set.
5. Click ⟳ to replace all of the members in the current set, then click **Apply and close**.
6. Click the view, click ▬ and 📊 , and select **Column**.
7. Click ⇤ to focus on the plan_BudgetPlan cube in the tree.
8. Right-click the ⊹ **plan_chart_of_accounts** dimension, and select **Add as selector widget**.

   The plan_charts_of_accounts dimension is added as a selector. You can size and position the dimension selector on the sheet.

   You can change the type of dimension selector to a slider. Click 📊 then select the type of dimension selector.
9. For the **Budget Input Detailed** view, click ▬. Then click ⌄ to hide the overview.

What to do next

Using the selector that you added, try selecting different items, and see how the chart changes.

Buttons

You can create a button in IBM Planning Analytics Workspace that lets you execute a TurboIntegrator process and/or open a new destination with a single click.

You can configure a button to:

- Execute a TurboIntegrator process
- Open a sheet in the current book
- Open a sheet in a different book
- Open a URL

When you configure a button to execute a TurboIntegrator process, you can also set the button to open a new destination after the process is complete.

Note: There are some differences between the behavior of buttons in Planning Analytics Workspace and buttons in Planning Analytics for Microsoft Excel/TM1 Websheets:

- In Planning Analytics Workspace, the execution of a TurboIntegrator process from a button does not lock the sheet or book. You can continue to work in the book or sheet while the process is running. In contrast, executing a process from a button in Planning Analytics for Microsoft Excel or a TM1 Websheet locks the sheet; you cannot perform any actions until the process finishes running.
- You can't use Excel cell references to supply process parameter values in Planning Analytics Workspace.

Procedure

1. To add a button to a sheet, you must be in edit mode. You are in edit mode if the pencil icon at the upper left of the screen looks like this icon . If necessary, click the pencil icon to enter edit mode.
2. Click the **Button** icon .

   If your sheet does not use a template (freeform), a new button is placed in the first open area of the sheet. If your sheet uses a template, the button is placed in the first open defined region of the template. You can drag the button to other locations of your sheet.
3. Use the handles on the perimeter of the button to resize the button as required.
4. Click the **Properties** icon , then click a category to set the display or action properties for the button.

   - **General style** - Sets the fill color, border color, and opacity of the button.
   - **Text properties** - Sets the style, font, alignment, and color of the button text.
   - **Button target** - Sets the process that is executed and/or the destination that is opened when you click the button.

     To execute an existing TurboIntegrator process:

     a. Select **Run process**.

     b. Select the database on which the process resides.

     c. Select the process you want to run.

     d. If the process includes parameters that must be set prior to execution, click **Set parameters** to set the parameter values or configure user prompts that can be answered to set parameter values.

     e. Select **Refresh after execution** if you want to refresh the current book after the process finishes running. The refresh updates any views, websheets, visualizations, or selectors in the target book.

     Select **Navigate to destination** to open a new destination when the button is clicked.

    To open a sheet in the current book, click **Sheet in this book**, then select the sheet you want to open from the list of available sheets.

    To open a sheet in a different book:

    a. Click **Other book**.
    b. Click **Select a book**.
    c. In the **Location** list, locate and open the folder containing the book you want to navigate to.
    d. In the **Books** list, click the book containing the sheet you want to open.
    e. In the **Tabs** list, click the tab/sheet you want to open.
    f. Click **Select**.
    g. Optionally, click **Pass context** to pass context for any items that have synchronization enabled in both the current sheet and the destination sheet.

    To open a URL, click **Hyperlink**, then enter a fully qualified URL.

  - **Button text** - Sets the text that appears on the button.

5. Click **Save** 💾.

Results

Users can now click the button to execute the process or move to the location you defined.

If the button executes a process, a visual progress indicator is displayed while the process is running. If the process runs successfully, the progress indicator closes, but you won't receive an explicit confirmation of process execution. If the process fails, you'll receive notification of failure directly in the sheet.

If you configure a button to navigate to a sheet in another book, you can rename the target book and/or sheet without having to update the button properties. The button automatically resolves any name changes to target sheets or books.

Set process parameters for a button

If you configure a button to execute a TurboIntegrator process that requires input parameters, you must configure questions that can be answered to provide parameter values or specify default parameter values.

About this task

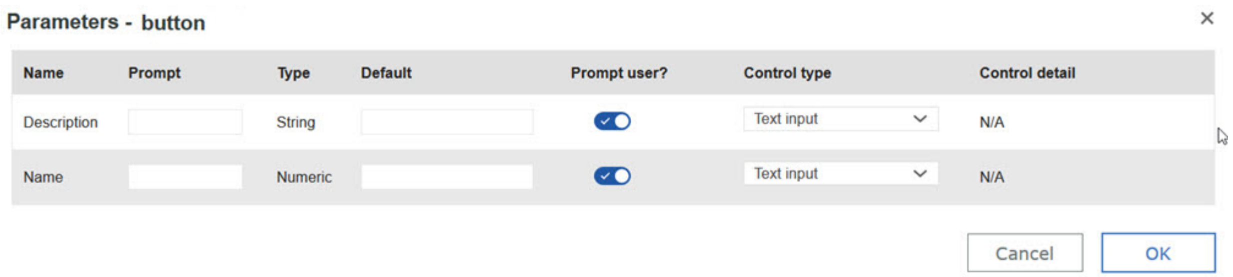
The names and types of parameters for a TurboIntegrator process are included in the process definition; these parameters are not defined by the button properties. However, when the process is executed from a button, values for the parameters must be passed into the process. You can configure a button to pass default parameter values into the process without requiring user input or you can configure prompts that ask the user to provide parameter values.

Procedure

1. To set process parameters for a button, you must be in edit mode.
2. After adding the button to a sheet, click the button to select it. You should see several sizing handles on the perimeter of the button indicating that it is selected.
3. Click the **Properties** icon, then click **Button target**.
4. Click **Set parameters**.

   If the process does not include any parameters, you'll see a message saying `No data available in table.`. In this case, you can click **Cancel** - you don't need to set any parameter values before running the process.

   Otherwise, you can see all the parameters that are include in the process on the Parameters screen.



Parameters - button

Name	Prompt	Type	Default	Prompt user?	Control type	Control detail
Description		String		✓	Text input	N/A
Name		Numeric		✓	Text input	N/A

The name and type for each parameter is defined in the process. You can't change these parameter properties, but knowing them helps you create a relevant prompt and set default values.

5. For each parameter:
   a) Enter a **Prompt** that the user will respond to when providing a parameter value. This should be a simple statement or question that clearly indicates what kind of response is required from the user.
   b) Set the **Prompt user** option. Select this option to display the prompt that you created when the process is executed from the button. The value that the user provides in response to the prompt is passed into the process. Deselect this option to pass the default parameter value directly into the process. Default parameter values are describe a bit later in this topic.
   c) Select a **Control type** and if necessary, set **Control detail**. These options determines how the user responds to the prompt to set the parameter value. The Control types available vary according to the parameter type.

   For string parameters, the following Control types are available:

   Text input
   Presents a box where the user can enter the parameter value. You do not need to specify any **Control detail**.

   List
   Presents a list of values from which the user can choose the parameter value. You must enter the values that you want to appear in the list in the **Control detail** column, with all values separated by commas.

   Dimension list
   Presents a list of dimension set members from which the user can choose the parameter value. To specify the dimension set, click **Select a dimension** and pick the dimension that contains the set you want to appear in the list. Then, click **Select a hierarchy** and pick the hierarchy that contains the set you want to use. Finally, click **Select a set** and pick the set containing the members you want to appear in the list.

   For numeric parameters, the following Control types are available:

   Text input
   Presents a box where the user can enter the parameter value. You do not need to specify any **Control detail**.

   List
   Presents a list of values that can be chosen to set the parameter value. You must enter the values that you want to appear in the list in the **Control detail** column, with all values separated by commas.

   Boolean
   Presents binary values that the user can choose. In the **Control detail** column, you can specify the **True label** and **False label**. The default values are `True` and `False`, but you can use `Yes` and `No`, `Yup` and `Nope`, or any labels you want.

   Number range
   Presents a box where the user can enter the parameter value. The value entered is validated against the range specified by the **From** and **To** limits in the **Control detail** column. If the value entered is outside the specified range, the user receives an error message.

Calendar
Presents a calendar from which the user can select a date value.

6. If necessary, enter a **Default** value that is appropriate for the Control type you're using. If you chose to not prompt users for parameter input, the default value is passed directly into the process when it is executed.

   Not all Control types require you to specify a default value, some derive the default value as outlined here:

   Text input
   Enter the default value you want to use.

   List
   Uses the first value in your comma-separated list as the default value. You can't modify this default value directly.

   Dimension list
   After you define the dimension set you want to let users to pick from in the **Control detail** column, click the **Pick an element** list in the Default column and pick the member you want to use as the default value.

   Boolean
   After you define a True label and False label, those labels appear in a menu on the Default column. Click the menu and pick the label you want to use as the default value.

   Number range
   Uses the From (lowest) value in number range as the default value. You can't modify this default value directly.

   Calendar
   When you specify a calendar Control type, a date picker 📅 is available on the Default column. Click the date picker and pick the date you want to use as the default value.

7. Click **OK** to save the parameter configuration.
8. While still in edit mode, you can click the button and then click ⇲ see how the button behaves and how the process runs.

Websheets

You can work with websheets in IBM Planning Analytics Workspace. You can create and work with sandboxes in websheets, and you can combine websheets and views on the same sheet.

A websheet is a Microsoft Excel worksheet (.xlsx file) with TM1 data that you can view in a web browser. Websheets are located in the **Applications** folder in the tree.

Note: In Planning Analytics Workspace, a websheet cannot display data from more than one TM1 database. This limitation applies only to websheets in Planning Analytics Workspace. In IBM Planning Analytics TM1 Web, websheets can display data from multiple TM1 databases that run on the same admin host and contain the credentials of the user account that is logged in to TM1 Web.

Procedure

1. To add a websheet to a book, you must be in edit mode. Click ✎ to enter edit mode.
2. Navigate to the **Applications** folder on the server that contains the websheet.

   For example, go to **Planning Sample** > **Applications** > **Planning Sample** > **Bottom Up Input**, and drag **Budget Input** onto the sheet.

3. To export websheets as PDFs or to Microsoft Excel, click the websheet, then click to open the toolbar, and click **Export**.
4. To reset data, in the toolbar, click **Reset data**.
5. To rebuild websheets and webbooks, in the toolbar, click **Rebuild websheet** or **Rebuild webbook**.

Save a websheet to the database

You can save an open websheet to the TM1 database.

About this task

When you save a websheet to the TM1 database, the websheet is available for quick retrieval from your collection.

A websheet saved in the database is a reference to the original websheet stored in the TM1 database. Any changes made to the original websheet in the TM1 database are propagated automatically to all associated websheets in the database.

Procedure

1. Click the websheet, then click to open the toolbar.
2. Click .

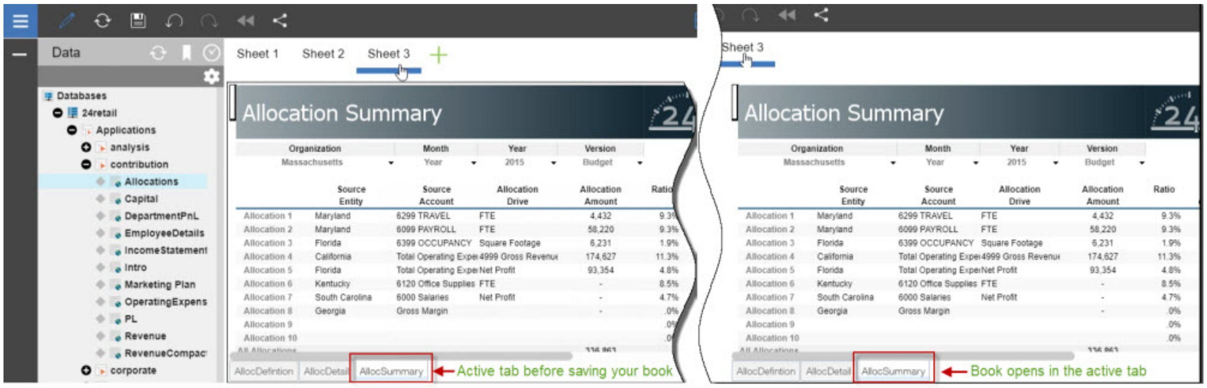
   The first time that you save a websheet, the only option available is **Save As**.
3. Click **Save As**, then provide a name for the websheet, and optionally, tags and a description that are applicable to the websheet.
4. After you initially save a websheet to the content store, you can use the **Save** or **Save as** options.

Results

The websheet is saved in your personal folder, from which you can open the websheet. You can also click and then click **Websheets** to access the websheet from your collection. You can't open the websheet directly from the collection, but you can drag the websheet from the collection on to an open Planning Analytics Workspace sheet.

Open a websheet on the active tab when you save a multi-tab websheet

If you have a websheet with multiple tabs in a book, Planning Analytics Workspace keeps track of the active websheet tab when you save the book. Then, when you open the book in Planning Analytics Workspace later, you are right where you left off in your work!



Organization	Month	Year	Version
Massachusetts	Year	2015	Budget

	Source Entity	Source Account	Allocation Drive	Allocation Amount	Ratio
Allocation 1	Maryland	6299 TRAVEL	FTE	4,432	9.3%
Allocation 2	Maryland	6099 PAYROLL	FTE	58,220	9.3%
Allocation 3	Florida	6399 OCCUPANCY	Square Footage	6,231	1.9%
Allocation 4	California	Total Operating Expe	4999 Gross Revenue	174,627	11.3%
Allocation 5	Florida	Total Operating Expe	Net Profit	93,354	4.8%
Allocation 6	Kentucky	6120 Office Supplies	FTE	-	8.5%
Allocation 7	South Carolina	6000 Salaries	Net Profit	-	4.7%
Allocation 8	Georgia	Gross Margin		-	.0%
Allocation 9					.0%
Allocation 10					.0%
All Allocations				336,863	

Using Planning Analytics version 2.0.9 and Planning Analytics Workspace version 2.0.46, when you open a book with a websheet in Planning Analytics Workspace, the tab that you saved the websheet with is active.

Note: You must be using Planning Analytics Workspace on IBM Planning Analytics version 2.0.9 to take advantage of this feature of websheets in IBM Planning Analytics TM1 Web.

If you don't have Planning Analytics version 2.0.9, the default tab that the websheet was published with (using TM1 Perspectives or Planning Analytics for Microsoft Excel) is active when you open the book in Planning Analytics Workspace.

Explore scenarios with sandboxes

Sandboxes let you try out different changes to the data before making those changes public to other users and before committing those changes to the base data. Sandboxes are visible only to you.

To use sandboxes, your administrator must set the **UseSandbox** capability to `Grant`. Capabilities can be set in the }Capabilities control cube.

Procedure

1. To create a sandbox, click the view or websheet to display the toolbar, and then click and click **Create sandbox**.
2. Name the sandbox.
3. Choose whether you want to create a sandbox from the base data or to create a copy of an existing sandbox, and then click **OK**.

   You can now work in the sandbox. You can move between different sandboxes by selecting them from the drop-down list. version 2 ▾
4. When you are satisfied with the data in a sandbox, and you want to commit it to the base, click ✓ **Commit data**.
5. To delete a sandbox, follow these steps.

   a) Click the view, and then click .

   b) Select **Delete sandbox**, select the sandbox that you want to remove, and click **Delete**.

What to do next

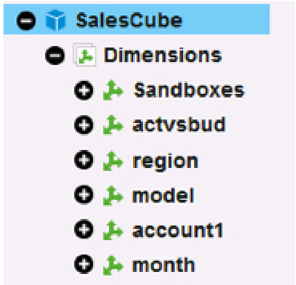
You can set sandboxes to synchronize, so that when a sandbox is set in one view or websheet, other views and websheets update to use the same sandbox. For details, see "Synchronize objects in a book or sheet" on page 116.

Compare sandboxes

You can create personal scenarios in sandboxes that you can view side by side so that you can compare and calculate the difference between scenarios.

Sandboxes are personal to the creator. If the compare sandbox function is enabled, and the Use Sandbox capability is granted by your administrator, you can use sandboxes as a virtual dimension in a view. An individual sandbox is treated as member in the virtual dimension.

You can see the **Sandboxes** dimension in the **Dimensions** branch in the tree, and as a tile that you can drag from ... . You can drag the **Sandboxes** dimension onto a row, column, or onto the context area.

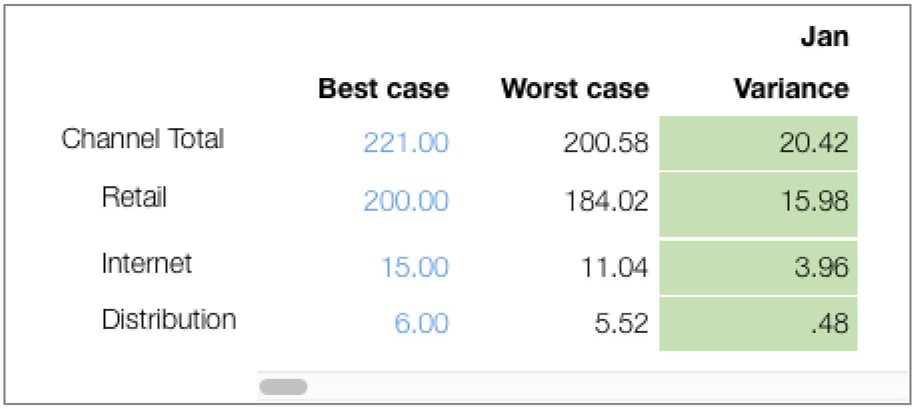


- SalesCube
  - Dimensions
    - Sandboxes
    - actvsbud
    - region
    - model
    - account1
    - month

You can also add the Sandboxes dimension as a selector. Right-click the **Sandboxes** dimension in the tree and select **Add as selector widget**. See "Create selectors" on page 152 to find out more.

Example

For example, suppose you have two sandboxes, Best case, and Worst case. You can display these two sandboxes next to each other in nested columns, and then calculate the variance, as shown in the following example.



			Jan
	Best case	Worst case	Variance
Channel Total	221.00	200.58	20.42
Retail	200.00	184.02	15.98
Internet	15.00	11.04	3.96
Distribution	6.00	5.52	.48

You can compare sandboxes by doing the following steps:

1. Drag the **Sandboxes** tile from ... , and nest it with the Month tile on columns.

   Tip: You can also drag **Sandboxes** from the **Dimensions** branch in the tree.
2. Right-click, or tap and hold on the Base member column heading and select **Hide**.
3. Create a variance calculation by selecting the Best case and Worst case column headers, right-clicking, and select **Create calculation**.

4. Name the calculation, choose **Best case - Worst case**, and click **OK**.

Limitations

You cannot spread data across multiple sandboxes.

You also cannot add members to the **Sandboxes** dimension. Sandboxes are added to the dimension when you create a new sandbox.

You can create unnamed sets for the sandbox dimension, and apply them to the current view, but you can't name and save sets.

Enabling the sandbox compare function

To enable the sandbox compare function, ask your TM1 administrator to add `EnableSandboxDimension=T` to the `tm1s.cfg` file of the TM1 database that you are using. For more information, see EnableSandboxDimension on IBM Knowledge Center.

Snap commands

Snap commands are simple commands that you can use to perform tasks quickly.

Click a view, then click ▼ to open the shortcut bar.

Next, click >_ and type a snap command.

You can type snap commands in full, or use abbreviations, for example. **swap** or **sw**. Snap commands aren't case-sensitive and you don't need to use exact spelling - `beljium` selects `Belgium`.

You can undo snap commands by clicking ↺.

For example, you can use snap commands to edit views as shown in this video.

https://youtu.be/hvXNuFmopAg

You can use these snap commands:

Select

Select members, sets, and levels in dimensions.

Type the names that you want to select in the snap command field that is separated by a `,` (comma), or and. Select is the default snap command, so you do not need to type `select`.

For example, type `germ,mark` to select the Germany and Marketing members.

Select doesn't apply to members on the bench ...

Swap

Swap rows, columns, dimensions in the context area. For example, to swap the versions and time dimensions, type `sw versions,time`. Typing swap on its own swaps the rows and columns.

Find

You can find members on rows, columns, or both.

`Find europe` finds the first result. To move through the results, type `find`. To go back, type, `find previous`.

Find is a powerful snap command, for example, typing `find r (color red or color blue) and size large` finds members on the rows, with an attribute value for color that is red or blue, and an attribute value for size that is large.

Hide

Hide named members in dimensions on the rows or columns.

To hide members that are named US and Feb-2004, type `hide US, Feb-2004`

Unhide

Show or unhide members.

Type `unhide r` to unhide rows. To unhide all, type `unhide`, or unh.

To unhide everything on the time hierarchy, type `unh time`

Keep

Keep members in dimensions on the rows or columns, while you hide the other members.

For example, type `keep 2014, 2015`

Zero

Enable zero suppression.

To enable zero suppression on just rows type `zero rows`, and for columns, type `zero columns`

To clear all zero suppression, type `zero off`

Totals

Show leading or trailing totals, or hide totals.

For example, in a time dimension, you could have Q1, Jan, Feb, Mar, which has a leading total. Typing `Totals trailing columns` would change the time dimension to: Jan, Feb, Mar, Q1.

Type `t -h` to hide all totals.

Level

Expand levels in a hierarchy to a specific depth. If you don't specify a level, the leaf level is expanded.

You can also expand numbered levels, and levels on rows, columns, or both. For example, type `level rows 2` or `level rows`.

Drill

Drill down on members.

For example, to drill down on a member named Georgia, type `Drill Georgia`.

Resetting the drill state clears drills. To reset the drill state for an Organization hierarchy, type `Drill reset Org`, or `d r org`

To remove all of the drills, type `drill reset`

Sort

Sort labels or values in ascending or descending order. You can choose to sort by hierarchy, or to break the hierarchy.

You can sort a named hierarchy, and for labels, you can sort by rows and columns. You don't need to specify label or value, unless there is an ambiguity.

Sort lab asc rows

Sorts labels in ascending order for rows, without breaking the hierarchy. You can also type this in natural language: `sort label ascending rows`.

Sort lab des br both

Sorts labels in descending order for both rows and columns, and breaks the hierarchy.

Sort val des br Sales

Sorts values in descending order, breaking the hierarchy, under the Sales member. You must specify a member to sort for values. In natural language, this is: `sort value descending breaking Sales`.

sort -x

Clears all sorting. To clear sorting on the months hierarchy, type `sort clear months`.

How the snap command searches

IBM Planning Analytics Workspace searches for member names, captions, and aliases, named sets, and named levels (levels with default names are ignored). If IBM Planning Analytics Workspace finds more than one member, they are listed in order of relative strength of the matches. More weight is given to precise matches than tentative matches.

Collaborate with chat

You can use the chat feature to collaborate with other IBM Planning Analytics Workspace users. Chat conversations are associated with and accessed from an individual book.

Any user with at least **View only** permission to a book can participate in chat. If you are an administrator, you can set permissions for a book.

When you log on to Planning Analytics Workspace, the chat icon 💬 on a book tile indicates whether there are any new messages or replies posted to the book.

https://youtu.be/xnCM00G5OIc

Procedure

1. With the book open in Planning Analytics Workspace, click 💬.
2. Click in the text box at the bottom of the chat panel, then enter your message and click **Post**.

   When you add links to web pages in a chat, they become clickable links. For example, type www.ibm.com into a chat, and the text becomes a link, colored blue. When you hover over the link, it becomes underlined. You can then click the link to go to the web page. Links beginning with www, http, or https are recognized as clickable links.

   Messages appear in reverse chronological order, with the most recent message at the top of the chat panel. When you post a message, your account avatar appears next to the message. If you have not set an account avatar, your first name and last name initials appear.
3. To reply to a message, click **Reply** (or **Replies** if the message already contains other responses), then enter your own message and click **Reply**.

   Chat supports a single level of replies. You can respond to a top-level message, but you can't respond to a reply.

   Replies appear in normal chronological order, with the oldest reply first and all other replies following sequentially.
4. To delete your own top-level message, click •••, then click 🗑.

Set your account avatar

You can upload an avatar image for your account. This avatar appears on the Planning Analytics Workspace dashboard next to your user name. The avatar also appears next to any messages you post in chat.

Procedure

1. Click your user name on the Planning Analytics Workspace dashboard.
2. Click **My Account**.
3. Click **Add Image**.
4. Find the image you want to use as your avatar on your local file system, then click **Open**.

   Files can be up to 500KB in size and can in .jpg, .gif, or .png format.
5. Click **Done**.

   To remove your avatar:
6. Click your user name on the dashboard.
7. Click **My Account**.
8. Click **Remove Image**.
9. Click **Done**.

Access your data and other objects quickly

You can quickly access your data and other objects in IBM Planning Analytics Workspace. You can add a view, a visualization, or objects to a collection to reuse them in a book. You can access recently used objects in the navigation tree. You can bookmark items in the navigation tree for later retrieval.

Save items in collections

When you create a useful view, visualization, selector, image, video, or other item, you can save the object in a collection for use in other books or sheets. You must be in Edit mode to save or retrieve an object from a collection.

Procedure

1. To save an item in a collection, click anywhere on the item, and click to open the shortcut bar. Then click .

   The item is saved to your collection.
2. To retrieve an item from a collection, click and then click **Collection**.
3. Click and hold the item, then drag it to a position on the open sheet.
4. To remove an item from your collection, click the item, then click **Remove**.
5. Click to close the collection.

Access recently used items

IBM Planning Analytics Workspace saves a list of recently used items so that you can quickly locate those items in the navigation tree.

Procedure

1. Click on the navigation tree.

   Tip: If the tree is not visible, click +.
2. Click the item that you want to locate in the **Recently visited** list.

   The item is selected in the navigation tree.

Bookmark items

You can bookmark items in the content tree for easy retrieval.

About this task

You can bookmark individual items, but not categories of items. For example, you can bookmark a specific cube, but not the entire Cubes category.

Procedure

1. To bookmark an item, right-click the item in the content tree, then tap **Bookmark this**.

   Tip: If the content tree is not visible, click +.
2. Click .
3. Click the bookmarked item that you want to locate in the content tree.

The item is selected in the content tree.

Export to Excel

You can export a cube view as a Microsoft Excel spreadsheet.

About this task

There are some size limitations that you need to be aware of. If you reach any of these limitations, you receive an error message.

Maximum number of rows
1048576

Maximum number of columns
16384

Maximum number of cells
8 million

Maximum number of view instances
100

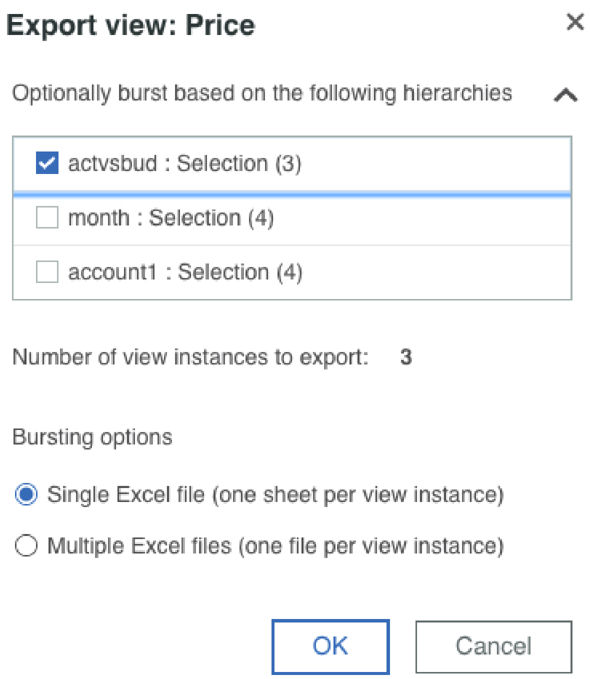
The total number of instances is determined by multiplying the number next to each option. If the number of instances is greater than 100, try changing your selection.

Export to Excel is not available from an iPAD.

Procedure

1. Click the cube view that contains the data that you want to export, then click to open the shortcut bar and click ⤒ on the shortcut bar.
2. Optional: Select which parts of the view that you want to export, based on hierarchy. Click ⌄ next to **Optionally burst based on the following hierarchies**.

   An option is shown for each dimension hierarchy that appears in the Context bar. If you select nothing, then the selection of data that is shown in the view is exported.



Export view: Price

Optionally burst based on the following hierarchies

☑ actvsbud : Selection (3)

□ month : Selection (4)

□ account1 : Selection (4)

Number of view instances to export: **3**

Bursting options

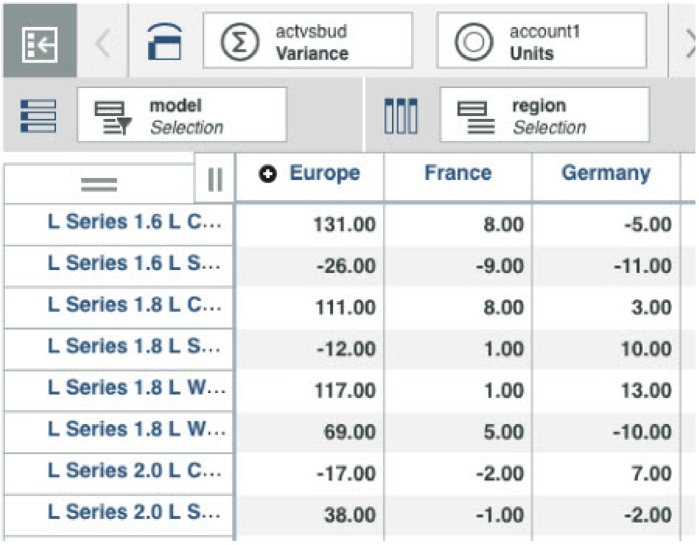
● Single Excel file (one sheet per view instance)

○ Multiple Excel files (one file per view instance)

3. Choose whether you want to export all sheets to a single file, or to multiple files. If you didn't choose to burst based on a hierarchy, only one sheet can be exported.
4. Click **OK** to begin the export process.

Example

Take as an example a view that shows models on the rows, and regions on the columns. On the context bar, Variance is selected in the actvsbud tile and Units is selected in the account1 tile. If you don't select a hierarchy to burst to, the view is exported as selected, with no additional sheets. But if you select a hierarchy, all possible combinations of views are exported.



actvsbud Variance

account1 Units

model *Selection*

region *Selection*

	Europe	France	Germany
L Series 1.6 L C...	131.00	8.00	-5.00
L Series 1.6 L S...	-26.00	-9.00	-11.00
L Series 1.8 L C...	111.00	8.00	3.00
L Series 1.8 L S...	-12.00	1.00	10.00
L Series 1.8 L W...	117.00	1.00	13.00
L Series 1.8 L W...	69.00	5.00	-10.00
L Series 2.0 L C...	-17.00	-2.00	7.00
L Series 2.0 L S...	38.00	-1.00	-2.00

For example, the actvsbud hierarchy has the following three items:

- Variance
- Actual
- Budget

If you select actvsbud hierarchy, data is exported to three sheets: Variance, Actual, and Budget.

The account1 hierarchy has three items at the same level:

- Units
- Sales
- Variable costs

If you select account1 hierarchy, data is exported to nine sheets:

- Variance_Units
- Variance_Sales
- Variance_Variable costs
- Actual_Units
- Actual_Sales
- Actual_Variable costs
- Budget_Units
- Budget_Sales
- Budget_Variable costs

Chapter 4. Use applications and plans to organize work

Applications and plans let you organize logically related Planning Analytics Workspace assets such as books, view, and websheets in containers.

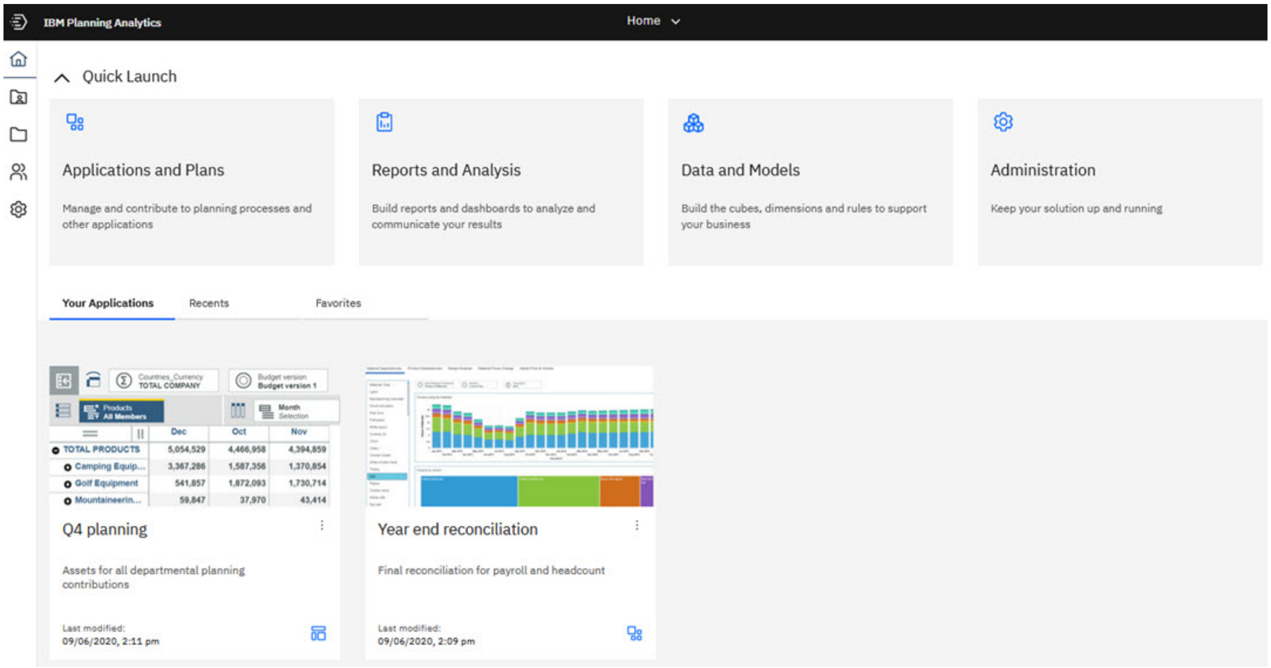
An application contains related assets that are grouped in sections. These sections might reflect the structure of your organization, planning and budgeting requirements, or any other relevant grouping of assets. While an application contains logically related assets, there are no implied or required actions associated with the assets or sections in an application.

A plan contains assets that are grouped in steps. These steps can represent discrete tasks or contributions that must be completed in a planning or budgeting process. While steps can be ordered in a plan, there is no requirement for contributors to complete the steps sequentially; they can be completed in any order. Steps can also be assigned a due date for contributions.

Access to sections in an application or steps in a plan are controlled by Planning Analytics Workspace user group assignments. When a user group is assigned to a section or step, the security for the contained assets is updated to include the user group.

Important: Application and plan security is applied only to Planning Analytics Workspace user groups and roles. There are no links or dependencies to TM1 object security. It is the responsibility of the Planning Analytics Workspace administrator to ensure that the Planning Analytics Workspace user groups have the requisite TM1 security permissions to view and edit assets.

When you log in to Planning Analytics Workspace, your applications and plans are visible on the **Your Application** tab at the bottom of the Home page. Applications are identified by the icon, while plans show the icon.



	Dec	Oct	Nov
TOTAL PRODUCTS	5,054,529	4,466,958	4,394,859
Camping Equip...	3,367,286	1,587,356	1,370,854
Golf Equipment	541,857	1,872,093	1,730,714
Mountaineerin...	59,847	37,970	43,414

Q4 planning

Assets for all departmental planning contributions

Last modified:
09/06/2020, 2:11 pm

Year end reconciliation

Final reconciliation for payroll and headcount

Last modified:
09/06/2020, 2:09 pm

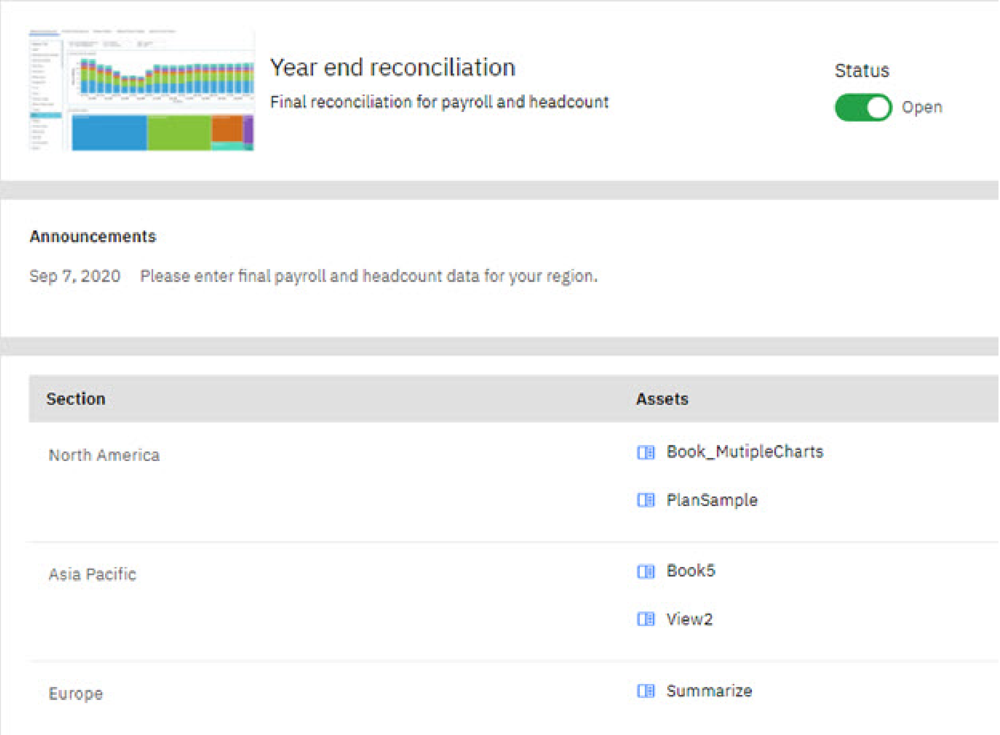
Administrators can see all applications and plans on the Home page, regardless of state or group assignments. Modelers, analysts and consumers can see open applications or plans in which they are a member of at least one of the user groups assigned to the included assets.

Applications

An application contains related assets, such as books, views, and websheets, that are grouped in sections. These sections might reflect the structure of your organization, planning and budgeting requirements, or any other relevant grouping of assets. While an application contains logically related assets, there are no implied or required actions associated with the assets or sections in an application.

Applications are created by a Planning Analytics Workspace administrator. When you log in to Planning Analytics Workspace, the applications that you can use appear at the bottom of the Home page on the Your Applications tab. Applications are identified by the ⊞ icon.

You can view the details of an application from either the Home page or the Applications and Plans page. The details page shows the status, announcements, sections, and assets for an application.



Year end reconciliation

Final reconciliation for payroll and headcount

Status

Open

Announcements

Sep 7, 2020 Please enter final payroll and headcount data for your region.

Section	Assets
North America	Book_MutipleCharts PlanSample
Asia Pacific	Book5 View2
Europe	Summarize

While the details page is largely a 'read-only' snapshot of an application, you can click any asset to open the full contribution page for the application.

To view the details of an application:

- If you are an administrator, click the **Options** icon ⋮ on the application tile, then click **Details**. You can also click the **Options** icon ⋮ for the application on the **Applications and Plans** page, then click **Details**.
- For all other users, click the application tile on the Home page to open the plan details. You can also click the application name on the **Applications and Plans** page to view application details.

Create an application

Create an application to organize your books, views, and websheets.

About this task

You must be a Planning Analytics Workspace administrator to create an application.

Any asset you want to include in an application must reside in the Planning Analytics Workspace Shared folder.

Procedure

1. On the Planning Analytics Workspace Home page, click **Applications and plans**.
2. On the Applications and Plans page, click **Create**, then click **Application**.

   A New Application template opens.
3. Click the **New application** placeholder text and enter a name for your application.
4. Click the **You can add a description here** placeholder text and enter a description for your application. The description you provide is visible on the application tile on the Planning Analytics Workspace Home page.
5. Click the **Logo** box and select a logo for the application. You can also drag and drop a logo onto the **Logo** box.
6. On the **Application details** tab, click **Click to add section**.
7. Click the **Untitled** placeholder and enter a name for the section.
8. Click **Assign assets**.

   The **Assign assets** page shows the books, views, and websheets that are available to add to your application section.

   - 📖 - Books
   - 👓 - Views
   - 📄 - Websheets

   You can search asset names or filter by asset type to narrow the list of assets.
9. Select the assets that you want to include in the current application section step, then click **Save**.

   The assets are added to the section in the order you selected them on the **Assign assets** page. You can click and drag assets to different positions to change the ordering.
10. Click **Assign groups**.

    The **Assign groups** page shows the existing user groups that you can assign to the section assets.

    Important: Application security is applied only to Planning Analytics Workspace user groups. There are no links or dependencies to TM1 object security. It is the responsibility of the Planning Analytics Workspace administrator to ensure that the Planning Analytics Workspace user groups have the requisite TM1 security permissions to view and edit assets.
11. Select the user groups that you want to assign to the section assets, then click **Save**.

    When a user group is assigned to a section, the security for the section assets is updated to provide access for the user group.
12. Repeat steps 6 through 11 for each additional step you want to add to the application.
13. Click the **Announcement** tab.
14. Click **Create**, then enter an announcement for the application.

    Announcements appear on the application contribution page and can be used to provide general instructions and other application details for your contributors. An application can contain multiple announcements. When multiple announcements are present, they are displayed in reverse chronological order, with the most recent announcement appearing first.
15. Click the **Open/Closed** application Status switch and select **Open** to open the application and make it available to users.

Manage an application

You can modify virtually any component of an application. You can add, remove, or reorder sections and assets in an application. You can also open or close the entire application, modify group assignments, and modify announcements.

About this task

You must be a Planning Analytics Workspace administrator to edit or manage an application.

Important: When you perform any action that affects asset or group security for an application you must explicitly refresh security by clicking **Actions**, then **Refresh Permissions**. Actions that require an explicit security refresh include:

- Adding an asset to a section
- Removing an asset from a section
- Adding a user group to a section
- Removing a user group from a section

Procedure

1. There are two ways to open an application for management:
   a) While logged in as an administrator, click the application tile on the Planning Analytics Workspace Home page.
   b) While logged in as an administrator, click the application name on the **Applications and Plans** page.
2. To open or close an application, click the application **Status** option to toggle the status of the application.
3. To modify announcements, click the **Announcements** tab.
   a) Click **Create** to add an announcement.
   b) Select an existing announcement and click **Delete** to remove an announcement.
4. To open or close an individual section, click the **Status** option to toggle the status of the step.
5. To rename a section, select the current section name, then type a new name.
6. To modify guidance for a section, click the **Guidance** icon ✎ and enter your changes.
7. To change the order of sections in an application, click just below a section name and then drag the section to a new position.
8. Modify section assets:
   a) To change the order of assets in a section, click an asset and drag it to a new position.
   b) To add assets to a section, click **Assign assets**, then select the assets and click **Save**.
   c) To remove an asset from a section, click the **Remove** icon × next to the asset name.

   Remember to click **Actions**, then **Refresh permissions** after you add or remove assets.
9. Modify group assignments for a section:
   a) To add user groups to a section, click **Assign groups**, then select the groups and click **Save**.
   b) To remove a user group from a section, click the **Remove** icon × next to the group name.

   Remember to click **Actions**, then **Refresh permissions** after you add or remove user groups.
10. To add a section to an application, click **Add a section**, then define the section as described in “Create an application” on page 168.
11. To delete a section from an application, click the **Delete** icon, then confirm the deletion.

Convert an application to a plan

When you convert an application to a plan, you can organize your assets in steps, assign due dates, and require contributor submissions.

About this task

You must be a Planning Analytics Workspace administrator to convert an application to a plan.

When you convert an application to a plan, the following actions happen:

- The new plan is created with a Closed status.
- The application logo is retained as the plan logo.

- All application announcements are removed.
- All application sections are converted to plan steps, and the status for each step is Closed.
- The section names from the application are retained and applied to the steps in the plan.
- Any section guidance is retained and applied to the steps in the plan.
- The assets and user groups from the application sections are applied to the steps in the plan.
- There are no due dates set for any plan steps.
- The **Require submission** option is set to **No** for all plan steps.

Procedure

1. Open the application management view. There are two ways to open an application for management:
   a) While logged in as an administrator, click the application tile on the Planning Analytics Workspace Home page.
   b) While logged in as an administrator, click the application name on the **Applications and Plans** page.
2. Click **Actions**, then **Save as**.
3. Enter a name for the plan, then select the **Plan** option and click **OK**.

Results

The plan opens in the plan management page. See "Manage a plan" on page 174 for details on modifying the plan.

Open and use an application

Any user that is a member of a user group that is assigned to at least one application asset can open and contribute to a plan.

About this task

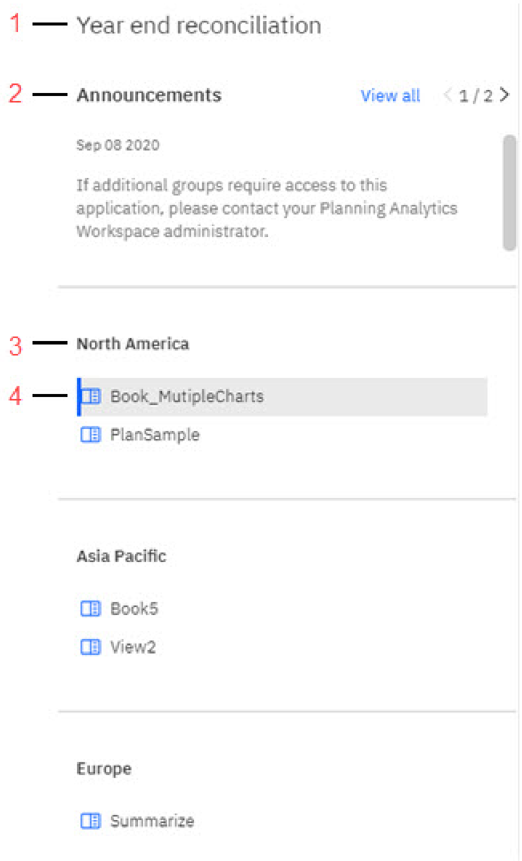
When you open an application, you can see and open only the sections and assets that have been assigned to a user group that you belong to. If an application contains five sections, but your user group has been assigned to three section, you'll see only those three sections.

Procedure

1. There are two ways to open an application for contribution:
   a) On the Home page, click the **Options** icon ⋮ on the application tile, then click **Open contribution**.
   b) On the Applications and Plans page, click the **Options** icon ⋮ for the application, then click **Open contribution**.

   The application opens in the contribution page. The first asset from the first section to which you have access is open and ready for your contribution.

   The left pane of the contribution page shows the sections and other information you need to work with the application.



Sep 08 2020

If additional groups require access to this application, please contact your Planning Analytics Workspace administrator.

1

Application name

2

Application announcements. The announcements display in reverse chronological order, with the most recent announcement appearing first. Click **View all** to view all announcements or click the arrows to scroll through announcements.

3

Application sections. You can see all sections that your user group has been assigned to.

4

Section assets

2. Complete your contribution to the first asset.
3. Click the next asset you want to work on and complete your contribution. Repeat until you've completed all the contributions you want to make.

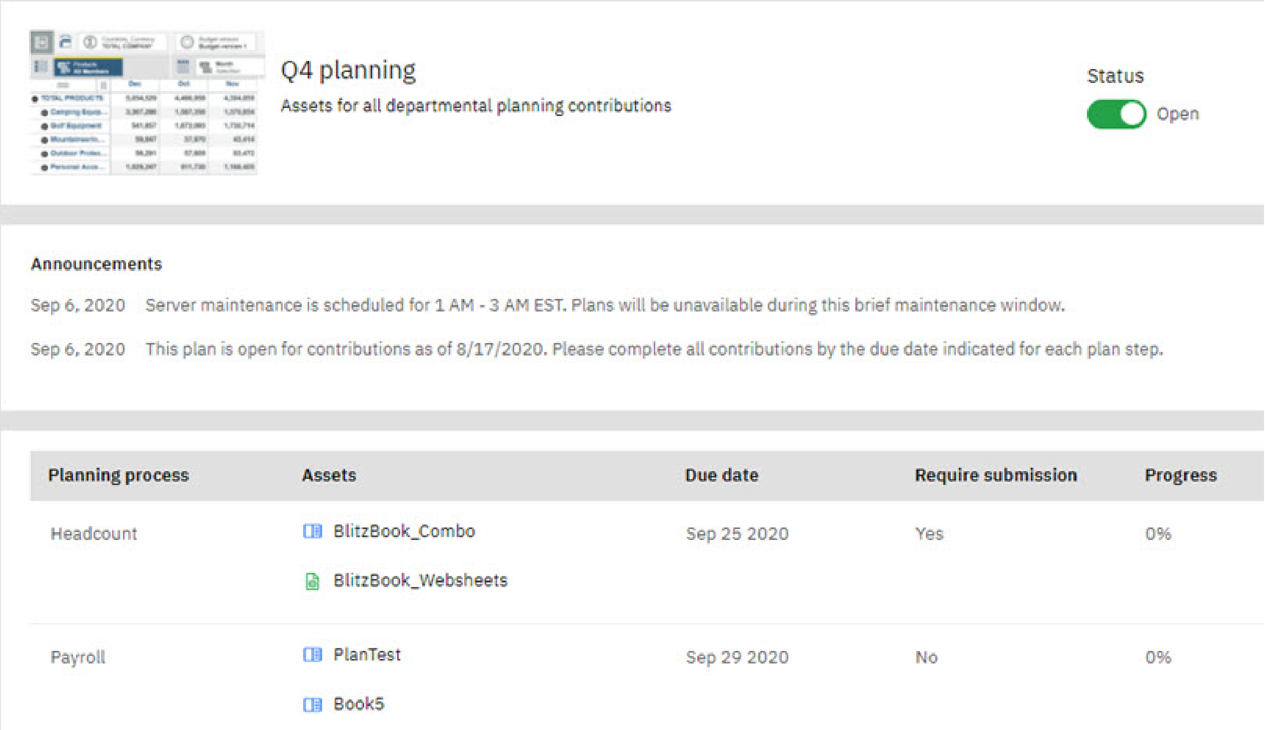
   You can work on sections or assets in any order.
4. When you're done, you can close the application from the Planning Analytics Workspace Home menu.

Plans

A plan contains assets that are grouped in steps. These steps can represent discrete tasks or contributions that must be completed in a planning or budgeting process. Each step in a plan must be completed by contributing data to the books, views, and websheteets in the step.

Plans are created by a Planning Analytics Workspace administrator. When you log in to Planning Analytics Workspace, the plans that you can use appear at the bottom of the Home page on the Your Applications tab. Plans are identified by the ▦ icon.

You can view the details of a plan from either the Home page or the Applications and Plans page. The details page shows the status, announcements, steps, and assets for a plan.



Q4 planning

Assets for all departmental planning contributions

Status

Open

Announcements

Sep 6, 2020 Server maintenance is scheduled for 1 AM - 3 AM EST. Plans will be unavailable during this brief maintenance window.

Sep 6, 2020 This plan is open for contributions as of 8/17/2020. Please complete all contributions by the due date indicated for each plan step.

Planning process	Assets	Due date	Require submission	Progress
Headcount	BlitzBook_Combo BlitzBook_Websheets	Sep 25 2020	Yes	0%
Payroll	PlanTest Book5	Sep 29 2020	No	0%

While the details page is largely a 'read-only' snapshot of a plan, you can click any asset to open the full contribution page for the plan.

To view the details of a plan:

- If you are an administrator, click the **Options** icon ⋮ on the plan tile, then click **Details**. You can also click the **Options** icon ⋮ for the plan on the **Applications and Plans** page, then click **Details**.
- For all other users, click the plan tile on the Home page to open the plan details. You can also click the plan name on the **Applications and Plans** page to view plan details.

Create a plan

Create a plan to organize the books, views, and websheeets used in your business processes and to provide guidance to process contributors.

About this task

You must be a Planning Analytics Workspace administrator to create a plan.

Any asset you want to include in a plan must reside in the Planning Analytics Workspace Shared folder.

Procedure

1. On the Planning Analytics Workspace Home page, click **Applications and plans**.
2. On the Applications and Plans page, click **Create**, then click **Plan**.

   A New Plan template opens.
3. Click the **New plan** placeholder text and enter a name for your plan.
4. Click the **You can add a description here** placeholder text and enter a description for your plan. The description you provide is visible on the plan tile on the Planning Analytics Workspace Home page.
5. Click the **Logo** box and select a logo for the plan. You can also drag and drop a logo onto the **Logo** box.
6. On the **Plan details** tab, click **Add a step**.
7. Click the **Untitled** placeholder and enter a name for the step.
8. Click the **Guidance** icon ✎.

9. Enter guidance or instructions for this specific step in your plan, then click **OK**.
10. Click **Assign assets**.

    The **Assign assets** page shows the books, views, and websheets that are available to add to your plan step.

    - 📖 - Books
    - 👓 - Views
    - 📄 - Websheets

    You can search asset names or filter by asset type to narrow the list of assets.
11. Select the assets that you want to include in the current plan step, then click **Save**.

    The assets are added to the plan step in the order you selected them on the **Assign assets** page. You can click and drag assets to different positions to change the ordering.
12. Click **Assign groups**.

    The **Assign groups** page shows the existing user groups that you can assign to the step assets.

    Important: Plan security is applied only to Planning Analytics Workspace user groups. There are no links or dependencies to TM1 object security. It is the responsibility of the Planning Analytics Workspace administrator to ensure that the Planning Analytics Workspace user groups have the requisite TM1 security permissions to view and edit assets.
13. Select the user groups that you want to assign to the step assets, then click **Save**.

    When a user group is assigned to a step, the security for the step assets is updated to provide access for the user group.
14. Click 📅 to set a due date for the plan step. The due date indicates the deadline for contributions to the plan step, but it does not have any impact on the state of an open step. When the due date is reached or surpassed, an open step remains open until it is closed by an administrator.
15. If you want the step to require recorded submission, click **Yes** under **Require submission**. Click **No** if recorded submission is not required.

    When a step requires submission, at least one user from one of the assigned user groups must click the **Submit** button on the plan contribution page to affirm that work has been completed for the step.
16. Click the **Open/Closed** status switch for the step and select **Open**.
17. Repeat steps 6 through 16 for each additional step you want to add to the plan.
18. Click the **Announcement** tab.
19. Click **Create**, then enter an announcement for the plan.

    Announcements appear on the plan contribution page and can be used to provide general instructions and other plan details for your contributors. A plan can contain multiple announcements. When multiple announcements are present, they are displayed in reverse chronological order, with the most recent announcement appearing first.
20. Click the **Open/Closed** plan Status switch and select **Open** to open the plan and make it available to contributors.

Manage a plan

You can modify virtually any component of a plan. You can add, remove, or change the order of steps and assets in a plan. You can modify group assignments, due dates, and submission requirements. You can also open or close the entire plan or individual plan steps. If your plan includes steps that require submission, you can manage group submissions as well.

About this task

You must be a Planning Analytics Workspace administrator to edit or manage a plan.

Important: When you perform any action that affects asset or group security for a plan you must explicitly refresh security by clicking **Actions**, then **Refresh Permissions**. Actions that require an explicit security refresh include:

- Adding an asset to a step
- Removing an asset from a step
- Adding a user group to a step
- Removing a user group from a step

While managing a plan, it's possible that other users might contribute submissions or otherwise modify the plan. You can click **Actions**, then **Refresh plan** at any time to update the plan management page.

Procedure

1. There are two ways to open a plan for editing or management:
    a) While logged in as an administrator, click the plan tile on the Planning Analytics Workspace Home page.

    b) While logged in as an administrator, click the plan name on the **Applications and Plans** page.
2. To open or close a plan, click the **Plan Status** option to toggle the status of the plan.
3. To modify plan announcements, click the **Announcements** tab.
    a) Click **Create** to add an announcement.

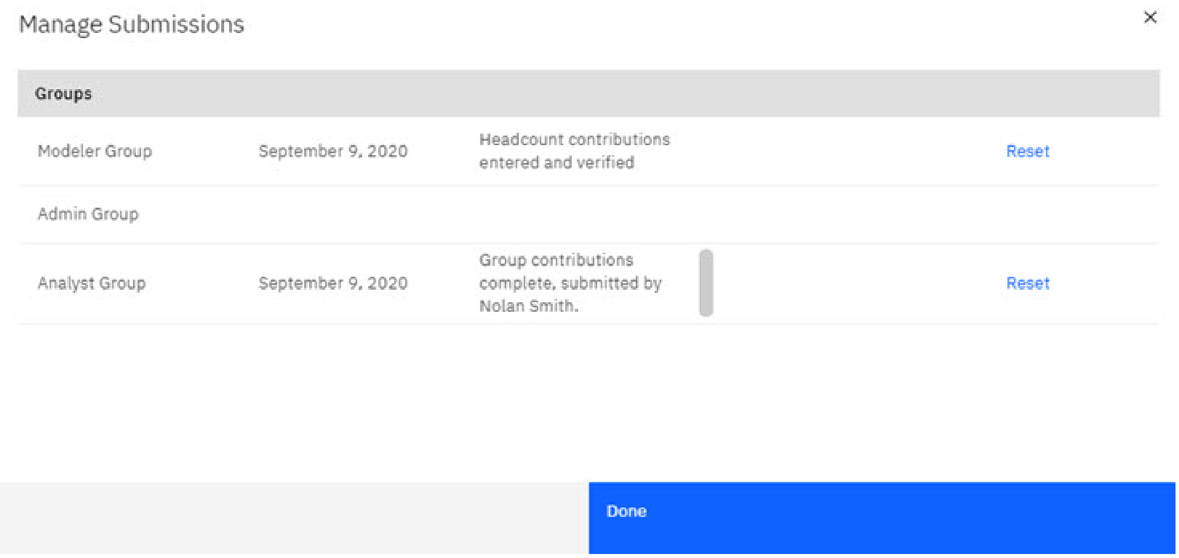
    b) Select an existing announcement and click **Delete** to remove an announcement.
4. To open or close an individual step, click the **Status** option to toggle the status of the step.
5. To rename a step, select the current step name, then type a new name.
6. To modify guidance for a step, click the **Guidance** icon ✎ and enter your changes.
7. To change the order of steps in an plan, click just below a step name and then drag the step to a new position.
8. Modify step assets:
    a) To change the order of assets in a step, click an asset and drag it to a new position.

    b) To add assets to a step, click **Assign assets**, then select the assets and click **Save**.

    c) To remove an asset from a step, click the **Remove** icon × next to the asset name.
9. Modify group assignments for a step:
    a) To add user groups to a step, click **Assign groups**, then select the groups and click **Save**.

    b) To remove a user group from a step, click the **Remove** icon × next to the group name.
10. To change the due date for a step, click 📅 and select a new date.
11. To change the submission requirement for a step, click the appropriate option under **Require submission**.
12. If a step requires submission, click **Manage Submissions** to view a report of group submissions for the step.

The **Manage Submissions** page shows the date on which each group submitted their contributions, along with any message that the submitter provides.



Manage Submissions

Groups			
Modeler Group	September 9, 2020	Headcount contributions entered and verified	Reset
Admin Group			
Analyst Group	September 9, 2020	Group contributions complete, submitted by Nolan Smith.	Reset

If the submission is not satisfactory, you can click **Reset** to rest the submission status for any group. The group must then contribute a revised submission. When you reset a submission, groups and users are not notified, but the submission status message for the group is removed from the plan contribution page and the status icons for the step are reset.

13. To add a step to a plan, click **Add a step**, then define the step as described in "Create a plan" on page 173.
14. To delete a step from a plan, click the **Delete** icon, then confirm the deletion.

Convert a plan to an application

When you convert a plan to an application, you retain all the assets and groups from the plan, but any artifacts related to due dates and submission requirements are removed.

About this task

You must be a Planning Analytics Workspace administrator to convert an application to a plan.

When you convert an application to a plan, the following actions happen:

- The new application is created with a Closed status.
- The plan logo is retained as the application logo.
- All announcements from the plan are removed.
- All plan steps are converted to application sections.
- The step names from the plan are retained and applied to the sections in the application.
- Any step guidance is retained and applied to the sections in the plan.
- The assets and user groups from the plan steps are applied to the section in the application.
- You cannot set due dates for application sections. All due dates from the plan are removed.
- You cannot require submission for application sections. All submission requirements from the plan are removed.

Procedure

1. Open the plan management view. There are two ways to open a plan for management:
   a) While logged in as an administrator, click the plan tile on the Planning Analytics Workspace Home page.
   b) While logged in as an administrator, click the plan name on the **Applications and Plans** page.

2. Click **Actions**, then **Save as**.
3. Enter a name for the application, then select the **Application** option and click **OK**.

Results

The new application opens in the application management page. See "Manage a plan" on page 174 for details on modifying the application.

Open and contribute to a plan

Any user that is a member of a user group that is assigned to at least one plan asset can open and contribute to a plan.

About this task

When you open a plan, you can see and open only the steps and assets that have been assigned to a user group that you belong to. If a plan contains five steps, but your user group has been assigned to three steps, you'll see only those three steps.

Procedure

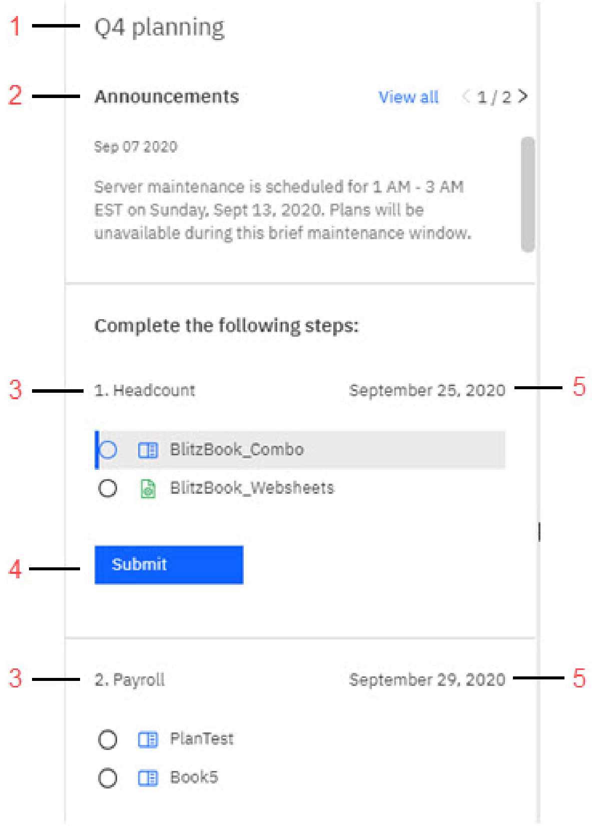
1. There are two ways to open a plan for contribution:

   a) On the Home page, click the **Options** icon ⋮ on the plan tile, then click **Open contribution**.

   b) On the Applications and Plans page, click the **Options** icon ⋮ for the plan , then click **Open contribution**.

   The plan opens in the contribution page. The first asset from the first step to which you have access is open and ready for your contribution.

   The left pane of the contribution page shows the steps and other information you need to work with the plan.



Sep 07 2020

Server maintenance is scheduled for 1 AM - 3 AM EST on Sunday, Sept 13, 2020. Plans will be unavailable during this brief maintenance window.

1

Plan name

2

Plan announcements. The announcements display in reverse chronological order, with the most recent announcement appearing first. Click **View all** to view all announcements or click the arrows to scroll through announcements.

3

Plan steps. You can see all steps that your user group has been assigned to.

4

Step due date

5

Step assets

6

Submit button. Steps in a plan can be configured by an administrator to require submission. If a step requires submission, the Submit button will be visible.

2. Complete your contribution to the first asset.
3. Click the next asset you want to work on and complete your contribution. Repeat until you've completed all your required contributions.

   Though steps are numbered sequentially, you can work on steps or assets in any order.
4. If a step requires explicit submission, click the **Submit** button after you've finished working on all assets in the step.
5. If you are a member of multiple groups that are assigned to the step, click the **Select** list and choose the user group on whose behalf you are submitting your plan contributions.

   If you are a member of only one user group assigned to the step, or if just a single user group is assigned to the step, or you can't choose a user group; the submission is made on behalf of your user group. You can also optionally enter a message to accompany your submission. The message can be viewed by your Planning Analytics Workspace administrator while managing submissions.
6. Click **Submit** to confirm the submission.

   When a step requires submission, the circle icon next to the step assets indicates the submission status for the step.

   - ○ - No submissions for any assets in the step.
   - ◒ - At least one user group has made a submission for at least one asset in the step.
   - ✓ - All user groups have completed submissions for all assets in the step.

   When your submission is complete, a status message displays below the step assets indicating the time and date of your submission.
7. When you're done working, you can close the plan from the Planning Analytics Workspace Home menu.

Chapter 5. Explore scorecards

Scorecards reflect the strategic goals of an organization. Using scorecards, you can identify how well objectives are being met by comparing targets to actual results. Visual status indicators such as traffic lights, trend icons, and colors are used to help you to quickly evaluate performance.

Note: Scorecards are supported in Planning Analytics Workspace Classic only. You cannot use scorecards in the new user interface introduced in Planning Analytics Workspace 2.0.57 SC.

In IBM Planning Analytics Workspace, you can add existing scorecards to your books, and analyze data by selecting different time periods, metrics, and dimensions. You can also create visualizations from scorecards, such as impact diagrams and strategy maps.

You can explore scorecards in Planning Analytics Workspace with the GO_Scorecards sample. For more information, see "Tutorial: Working with scorecards" on page 360.

Scorecards

A scorecard is a collection of performance metrics that are designed to reflect the strategic goals of your business unit or organization.

The information in a scorecard identifies how well the objectives are being met by comparing planned to actual results. Scorecards can also show information for the different organizations in your business. By using visual status indicators such as traffic light, and trend icons, scorecards can help users to quickly evaluate performance.

A scorecard combines data and dimensions into interactive diagrams and visualizations that you can share with other users. Scorecards include the following elements:

Metrics
A measure or key performance indicator (KPI) that conveys the performance of an important area of your business. Examples include Profit, Revenue, and Expenses.

Metric indicator
A measure of performance, status, or trend for a key area (metric) of your business. A metric indicator compares current results to target values. For example, Score, Status, and Trend.

Impact diagram

Impact diagrams illustrate the positive and negative relationships between the metrics in your metrics cube. This type of diagram shows how the business works by displaying how one metric influences another.

An example for an impact diagram might show how Revenue and Expenses influence Profit, which then affects Bonuses and Research Funding.

Impact diagrams display traffic light and trend indicators that show the status and the trend of each metric in the diagram. You can filter for different contexts in the impact diagram. The traffic light and trend indicators update with new values for the selected dimension.

When you double-click a metric in an impact diagram, the selected metric become the focus metric. Views, visualizations, and other widgets on the current dashboard also update automatically to show data in context of the focus metric.

Strategy map

A strategy map is a visualization that tracks business performance by perspectives, objectives, and metrics.

A strategy map shows the status of metrics with traffic lights and trend indicator icons. A strategy map organizes perspectives, objectives, and metrics into the following hierarchy:

- A strategy map can have multiple perspectives.

- Each perspective can have multiple objectives.
- Each objective can have multiple metrics.

The standard perspectives for a strategy map include the following items:

- Financial performance
- Customer knowledge
- Internal business processes
- Learning and growth

Connections in a strategy map display as directional arrows to show a visual relationship or flow between the objectives in the diagram.

When you double-click a metric in a strategy map, the selected metric become the focus metric. Views, visualizations, and other widgets on the current dashboard also update automatically to show data in context of the focus metric.

Custom diagram

A custom diagram is a strategy map that has a custom image and shows metrics with dimensional context onto the image as data points.

These examples of custom diagrams are available:

Geographical maps
Show a regional focus of your organization.

Process diagrams
Shows metrics in the context of a process flow.

A custom diagram displays the metric and context dimension names with traffic light and trend indicator icons as an overlay or layer on the selected image.

Metrics cubes

A metrics cube is a special type of cube that provides the basis for scorecard solutions and scorecard diagrams.

A metrics cube monitors multiple metrics and metric indicators. The primary feature of a metrics cube is that it shows the current relative status of many rows in a table. It displays the current trend of many measures simultaneously.

The standard scorecard layout for a metrics cube is as follows:

- Row title dimension: metric dimension
- Column title dimension: metric indicator dimension
- Context dimensions: time, geography, and other data context dimensions

A metrics cube combines a metric dimension and metric indicator dimension with your other regular dimensions. Metrics cubes have the same properties of other cubes.

A metrics cube must contain the following dimensions as a minimum:

- A metric dimension
- A metric indicator dimension
- A time dimension

Metric dimension

The metric dimension contains your collection of important measures or key performance indicators (KPI) that you want to monitor in your business or organization.

These measures are called metrics and each identifies one aspect of performance, such as Gross profit, Revenue, or Product cost. You can monitor the actual performance of a metric and compare it to expected or target values by combining it with metric indicators to provide the additional details about status, score, and trend.

Metric indicator dimension

A metric indicator dimension provides more information about your key performance indicators (KPI) or metrics. Examples of metric indicators include Score, Status, and Trend.

The metric indicators measure the performance, status, and trends in key areas of a business by comparing current results to target values. For example, the Actual, Target, and Tolerance indicators for a metric are typically used to calculate the related Score, Status, and Trend indicators.

Traffic light status indicator

A traffic light status indicator shows if metrics are meeting set targets.

The status is indicated by the color and the shape of the icon.

Table 6. Metric indicator traffic light status icons

Traffic light icon	Description
🟢	A green circle icon indicates excellent status, with metric being on-target or better.
🔶	A yellow diamond icon indicates average status, with the metric being off-target but within an acceptable range.
🟥	A red square icon indicates poor status, with the metric off-target and unacceptable.
●■◆	This image indicates missing data that makes it impossible to compute status.

Trend indicator

A trend indicator shows how the value of one column compares to the value of another column.

Trend indicators convey whether performance is improving or getting worse, not going up or going down. You can determine performance at a glance without having to decide whether an increase is good or bad.

Table 7. Metric indicator trend icons

Trend icon	Description
⬆	A green upward facing arrow indicates that the trend value is improving in comparison to the previous period. For example, a sales value is greater than the previous month or quarter.
▬	A gray dash icon indicates that the trend value is unchanged.
⬇	A red downward facing arrow indicates that the trend value is worsening in comparison to the previous period. For example, a sales value is less than the previous month or quarter.
Blank cell	A blank cell indicates that the trend is missing data for that period. A trend cannot be displayed when there is an incomplete status. For example, a trend cannot be displayed for the first time period, such as Q1 (quarter one). Previous data does not exist, even if the metric has a value for Actual, Target, Score, and Status.

Chapter 6. Model in Planning Analytics Workspace

IBM Planning Analytics Workspace includes a modeling environment that you can use to model user data with cubes, dimensions, hierarchies, attributes, and security for IBM Planning Analytics.

OLAP modeling and modeling concepts can be confusing. This video will help you to understand some of the basic concepts of what they are and why we model data for analyzes:

https://youtu.be/OnE9MBUHfn0

Note: Planning Analytics Workspace modeling is supported on IBM Planning Analytics Local 2.0.0 or later. Modeling is not supported in Planning Analytics Workspace on TM1 databases that are installed with version 10.3.0 or earlier.

You can use the Planning Analytics Workspace modeling tools to convert business requirements into precise cubes, dimensions, hierarchies, and calculations so that planning and analytics outcomes make sense to business users.

- Build and maintain the structure of a financial model independently, without coding.
- Transform and load data easily into a financial model, which increases transparency and confidence in results.
- Build a step-by-step financial process for multiple users based on roles and security permissions, without coding.
- Modelers can define business logic by using an integrated development environment.

To use Planning Analytics Workspace modeling, you must log in with a user name that has the Modeler role.

Planning Analytics Workspace modeling supports the following tasks:

- Creating cubes
- Using rules
- Editing dimensions
- Managing hierarchies
- Creating attributes
- Managing security

To understand why you should use the Planning Analytics Workspace tools to model your data, consider the following benefits.

Query performance

You can create more manageable dimensions by creating more than one hierarchy in a dimension.

For example, you can create a single time dimension. A year dimension must be the same for every year to compare data between two years. This approach gives you the ability to create a new year easily, and query performance is faster because you have only one dimension in the cube.

You can create a time dimension with multiple hierarchies that represent years and months to do yearly comparisons per month.

If you create two dimensions, one for year and one for months, you gain the ability to split the years and months across two axes and you can compare data between years.

RAM savings

If you model with dimensions, you can put the dimensions on the cube axes. Dimensions describe the data, and you can query against them. But dimensions cost RAM memory. Therefore, you can use hierarchies to model your data instead of dimensions.

If you model with attributes, it keeps your cube structure simple but it describes the dimension members only, not the data itself. You can't use attributes to query data. Creating a dimension for attributes that you want to query on creates complexity.

You can use hierarchies to model your data and get the same granularity that attributes give you with the benefit of being able to query on the data. You can query directly with hierarchies on the axes or with set filtering. You save on storage costs and complexity if you model with hierarchies.

Flexibility

Using hierarchies to add versions creates flexibility. For example, you might need to change your organizational hierarchy for planned changes. One dimension might be the organization and you can use hierarchies in the dimension to represent the organization in the future year. This hierarchy might represent the data differently from the previous year's hierarchy. With multiple hierarchies that represent the organization, you can roll up the data in multiple ways.

Hierarchies are named and contain members. You can reuse the same consolidated members in multiple hierarchies. You can use hierarchies to group these members without the need for specific consolidated names.

Standards

Hierarchies conform to OLAP Industry standards. Planning Analytics Workspace modeling uses MDX and TM1 REST APIs to access TM1 data. The TM1 REST APIs support the hierarchy model and follow the ODATA standard.

The modeler role

If you are enabled as an Administrator or a Modeler in Planning Analytics Workspace, you can design dimensions, hierarchies, views, and attributes to define the business logic for your application.

Your role is set when you are added to Planning Analytics Workspace by an administrator. If you are enabled as a Modeler, you can use all of the capabilities of an Analyst plus the modeling capabilities. For more information about adding users and setting their roles, see "Administer users and groups" on page 267. To learn more about roles and capabilities, see "User roles" on page 31.

To use the modeling capabilities, log in to Planning Analytics Workspace with a user name that has a role of **Administrator** or **Modeler**.

When you log in to Planning Analytics Workspace from an iPad, you are always working in Consumer mode, therefore you cannot use the modeling capabilities. For more information, see "Accessing Planning Analytics Workspace from Apple iPad" on page 36.

Steps to building a model

You design a model in response to a business need. For example, a company wants to plan its expenses for the next 12 months by expense line. The company collects the information from a number of departments that are spread across a wide geographical area.

A model has the following basic building blocks: dimensions, cubes, and links.

Multidimensional cubes are central to the model. You create views from the cubes so that you can see just the information that you need, and you can have many different views of the same cube. For example, a regional manager might want to review data at a consolidated level, and a department manager might want to input the detailed data for their department. They use the same cube but see different views.

Dimensions give the cubes structure. A cube must have a minimum of two dimensions, but the total number of dimensions a cube can have is determined by its use. A cube that stores data can have many dimensions. A cube that is viewed by a user should have enough dimensions to define the data, but not so many dimensions that the cube is difficult to navigate. You can use hierarchies to reduce the number of dimensions in a model, while giving you the option to see alternative rollups of data.

You can quickly add business logic such as links, by creating rules.

You make the data available to other colleagues by creating books and adding views, scorecards, visualizations, graphics, and videos, and then sharing the book with them.

Design dimensions

To make the data available for input and analysis, you must first create dimensions. Some examples of dimensions are Chart of Accounts, Products, Time, and Versions.

Find out more in "Dimensions" on page 185.

Create hierarchies

Use hierarchies to simplify your model design. A hierarchy acts as a virtual dimension, enabling you to see alternative rollups of the same view without creating additional dimensions. This makes a model easier to maintain.

Find out more in "Hierarchies" on page 208.

Create cubes

Use dimensions to build cubes. A cube is a store of data within a model. It is multidimensional and contains rows, columns, and any number of pages. Some examples of cubes are sales planning or expense analysis.

Find out more in "Cubes" on page 214.

Create rules and processes

You can create rules for calculations, and processes for managing and maintaining the model. Processes can then be grouped into chores for ongoing maintenance.

Find out more in "Rules" on page 228, "TurboIntegrator processes" on page 248, and "TurboIntegrator chores" on page 259.

Dimensions

Dimensions are lists of related members. Two or more dimensions are used to make a cube that can be used for planning and analysis.

Typical dimensions a cube might contain are time, versions, regions, products, departments, measures. A member is an item in a dimension, so in a time dimension, you can have months, years, quarters. Each month, year, and quarter is a member.

Dimensions can be a simple list with all members at the same level, or a dimension can be structured with members at different levels and with multiple hierarchies. How a dimension is structured depends on how you want the data to be represented. You might want to have a simple time dimension that just contains a list of the months, or you might want a time dimension that is grouped by years, quarters, and months, as shown in the following list.

- 2017
  - Q1-2017
    - Jan-2017
    - Feb-2017
    - Mar-2017
  - Q2-2017
    - Apr-2017
    - May-2017
    - Jun-2017
  - Q3-2017

    - Jul-2017
    - Aug-2017
    - Sep-2017
  - Q4-2017
    - Oct-2017
    - Nov-2017
    - Dec-2017

Levels define the way data is grouped in dimensions. A dimension can have a number of levels relative to their hierarchical structure, and these levels are automatically named Level000, Level001, Level002, Level003 and so on. For a dimension with a single level, the level is named Level000.

If you have a dimension that is structured with multiple levels, you can choose to show the members at a particular level. For example, you could show the leaf level, which is just the months, or you could show just the quarters, or the years. To select a level, in a cube view, click the dimension tile and then select the level.

Typically, cubes contain a measures dimension. A measures dimension contains the measures that you want to track in your business analysis. Examples of measures include sales amounts, units sold, expenses, acquisition values, and campaign costs.

You can define hierarchies for dimensions. Every dimension in IBM Planning Analytics Workspace has at least one hierarchy. You can define alternative hierarchies if this feature is enabled so that you can roll up a hierarchy in different ways without having to add extra dimensions. To find out more, see "Hierarchies" on page 208.

Dimension attributes provide information about the dimension. You can view and edit the values for dimension attributes, but you cannot add them in the settings editor.

For more information, see "Dimension settings" on page 206.

You can define attributes for dimension members. Attributes help to explain or describe a dimension member, and could be something like color, size, or type. You can create hierarchies from attributes by right-clicking an attribute in a dimension and clicking **Create Hierarchy**.

Create a dimension

You can create and edit dimensions when you are a Modeler or an Administrator.

Procedure

1. In edit mode, in the **Data** tree, go to the database where you want to create the dimension.
2. Right-click **Dimensions**, and click **Create dimension**.
3. Enter the name of your dimension and click **Create**.

   The name can't contain these characters: `\ / : * ? " < > | ' ; , }`

   By default, a dimension is created with a hierarchy that has the same name as the dimension.
4. Add members to the dimension by using one of these methods:
   - Import members from a file ("Import members and attributes into a dimension" on page 187).
   - Copy and paste members into the dimension ("Add members to a dimension" on page 201).

   You can also customize the dimension as a Time dimension.

What to do next

To create another dimension, click ⋮ next to the dimension tile and click **Create dimension**.

To create a new hierarchy within the current dimension, click ⋮ next to your hierarchy tile and click **Create hierarchy**. For more information about creating hierarchies, see “Hierarchies” on page 208.

To delete a dimension, in the **Data** tree, right-click the dimension, and click **Delete dimension**.

Import members and attributes into a dimension

You can import dimension members, attributes, types, and member weight properties into a dimension. You can also create a process and a chore so that you can regularly update the dimension.

Your import data must be correctly formatted to get the right results. To find out more, see “Dimension import file formats” on page 189.

In IBM Planning Analytics Workspace Local, drag and drop requires Planning Analytics Administration agent to be configured and running.

Procedure

1. To create a new dimension, right-click **Dimensions**, and click **Create dimension**. Name the dimension, then either click **Browse for file**, or drag the import file onto the dimension editor window.
2. To update an existing dimension, click ⤓ in the dimension editor, and browse for the import file.

   A preview of the file that you are importing is displayed. The delimiter and quote characters that are used in the text file are automatically detected, and the **Header rows** setting defaults to 1. You can change these settings.

   If you change the settings, you must click refresh ⟳ before you can continue to the next step.
3. Click **Continue**.
4. Select the dimension import settings under **Mapping type** in the right pane.

   Leaf only

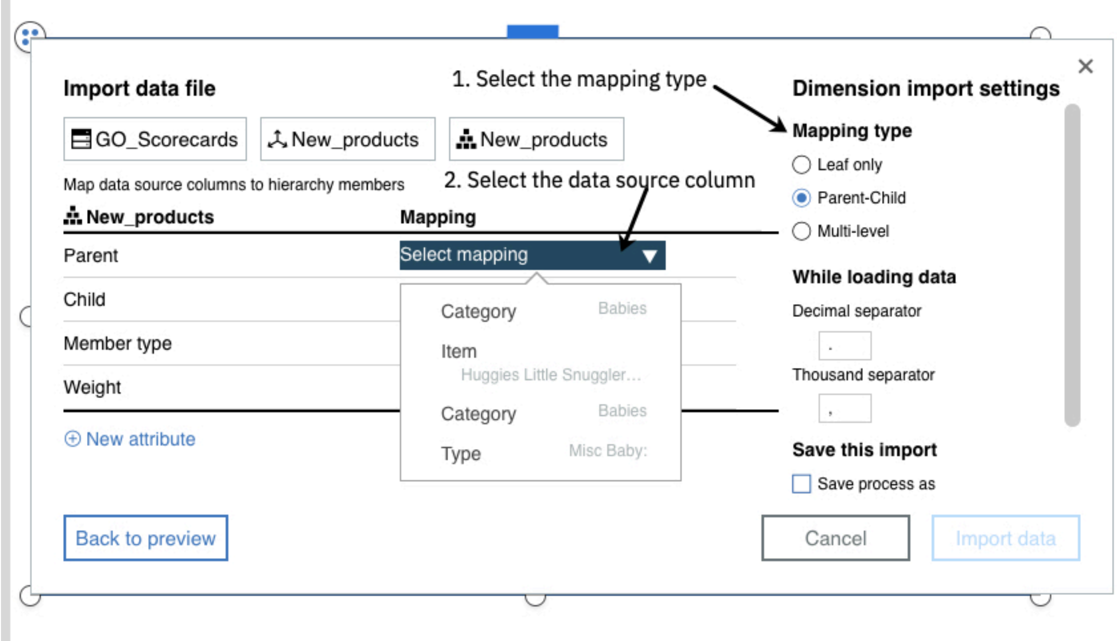
   Imports a flat list of members.

   Parent-Child

   Imports two columns of data with a parent-child relationship. For example, Scotland is the parent and Edinburgh is the child. Parent-Child is the default setting.

   Multi-level

   Creates more than two levels in a dimension.



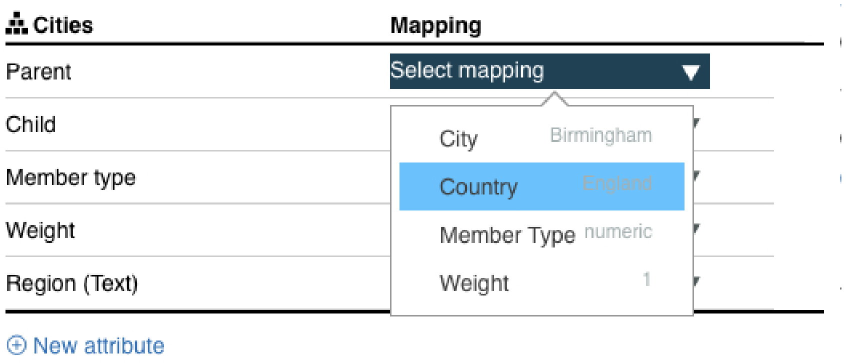
1. Select the mapping type

2. Select the data source column

5. Map your data source columns to the hierarchy members.

   In the **Mapping** list, the headers from the source data are listed, along with the first member.

   In the example shown, Country is the Parent.



   Leaf only
   Select the column of data you want to import as the leaf level under **Mapping**.

   Parent-Child
   Select the columns of data that you want to import as the **Parent** and the **Child** members.

   Multi-level
   Select the columns of data to import as the **Leaf**, **Level 1**, and **Level 2** members. If you need more levels, click **Add level**.

6. To set the **Member type**, in the **Member type** field, select either **Numeric** or **String** if all members have the same type, or select the column in the source file that contains the member type.

   Strings are used in pick lists, and in cells where users type comments.

7. To assign the same weight to all members in the dimension, type the weight value, such as 1.0 in the **Weight** field. To import weights from the source file, select the column that contains the weight values.

   Top-level consolidations do not have a weighting. To find out more about weights, see "Weights" on page 192.

8. To map an existing attribute to a column of data, select the column of data next to the attribute.

9. To create a new attribute, click **New attribute**, type the name of the attribute, and select the type: **Text**, **Number**, or **Alias**.

Then, click the arrow to map the appropriate source column to the attribute.

10. If you are importing into a dimension that contains members, select one of the following **Dimension import settings** under **While loading data**.

    - **Re-creates the current hierarchy**
    - **Update hierarchy with new members**

11. You can change the default **Decimal separator** and **Thousand separator**.
12. To create a process that contains the options that you specified, select the **Save as process** option, and name the process.

    The process is saved in the **Processes** branch of the content tree. You can modify the process, see "Create and edit processes" on page 248.

13. To schedule the process that you created, select the **Schedule process as** option, and name the chore.

    A chore is created and opened on the tab so that you can set the schedule, see "Create and edit chores" on page 259. The chore is saved in the **Chores** branch of the content tree.

14. Click **Import**.

Dimension import file formats

When you import a file into a dimension, the file must be structured and formatted in a certain way to get the results that you want.

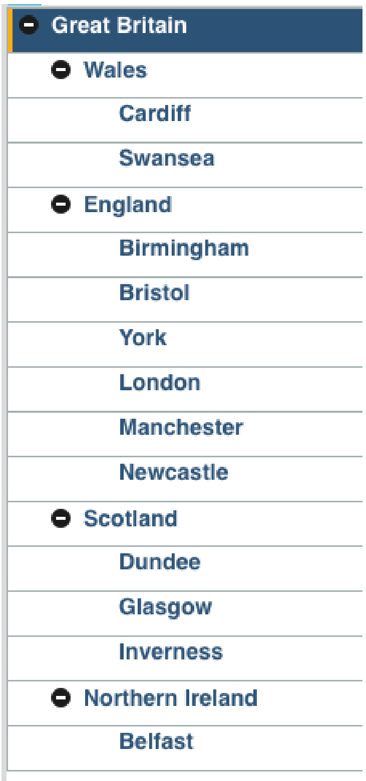
File formats

The import data must be saved in a format, such as comma-separated (.csv) or with any other standard delimiter such as tab, space, semicolon, colon, vertical line, or caret (^).

Note: Files that contain a Byte Order Marker (BOM) might be corrupted when the file is imported.

Parent-child dimension structure

If you want to create a dimension with a parent-child structure, the import file must have two columns, and the levels are defined in a parent-child relationship. Any extra columns are ignored. Headings are optional. A member can be both a child and a parent; the members Wales, England, Scotland, and Northern Ireland are both children and parents. An example of a dimension that was created by importing a file that is structured to have parent-child relationships is shown in the following picture.



- Great Britain
  - Wales
    - Cardiff
    - Swansea
  - England
    - Birmingham
    - Bristol
    - York
    - London
    - Manchester
    - Newcastle
  - Scotland
    - Dundee
    - Glasgow
    - Inverness
  - Northern Ireland
    - Belfast

The example can be created with a .csv file that contains either of the following formats.

Column-separated

The first column contains the child dimension members; the second column contains the parent dimension members.

Column 1	**Column 2**
Belfast	Northern Ireland
Birmingham	England
Bristol	England
Cardiff	Wales
Dundee	Scotland
Glasgow	Scotland
York	England
Inverness	Scotland
London	England
Manchester	England
Newcastle	England
Swansea	Wales
Wales	Great Britain
England	Great Britain
Scotland	Great Britain
Northern Ireland	Great Britain

Comma-separated

The value preceding the comma contains the child dimension members and the value after the comma contains the parent dimension members.

```
Belfast, Northern Ireland
Birmingham, England
Bristol, England
Cardiff, Wales
Dundee, Scotland
Glasgow, Scotland
York, England
Inverness, Scotland
London, England
Manchester, England
Newcastle, England
Swansea, Wales
Wales, Great Britain
England, Great Britain
Scotland, Great Britain
Northern Ireland, Great Britain
```

Tip: Some text editing applications may add additional metadata to your .csv file, which can affect the import process. For the best results, use a simple text editor for data that's comma-separated.

Importing attributes, weights, and member type

You can import attributes, weights, and member types alongside members. Your file should have a column of data for the attribute, weight, and member type. Weight can only be imported for parent-child dimension structures.

The following example has 4 columns of data, with Family as the Parent, Model as the child, Engine Size as attribute, and Weight. You can import multiple attributes, and you can import into existing attributes, or you can create a new attribute during the file import.

```
Type,Model,Engine Size, Weight
Budget,L Series 1.8 L Sedan,1.8,1
Budget,L Series 2.0 L Sedan,2,1
Budget,L Series 2.5 L Sedan,2.5,1
Budget,S Series 1.8 L Sedan,1.8,1
Family,L Series 1.8 L Wagon,1.8,1
Family,L Series 1.8 L Wagon 4WD,1.8,1
Family,L Series 2.0 L Wagon,2,1
Family,L Series 2.0 L Wagon 4WD,2,1
Family,S Series 3.0 L Wagon 4WD,3,1
```

The following example has 2 columns of data. The first column is a leaf member, and the second column is the member type. The member type can be string (s), or number (n).

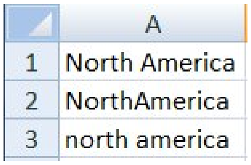
```
Name/Desc,s
Job Type,s
Job Code,s
FTE,n
50P Salary,n
Current Salary,n
Merit Pd,s
```

Member names

The member names that are contained in the source files should follow "Naming conventions" on page 644.

If the source file has multiple occurrences of a member name, the member is created by using the first occurrence of the member name.

You can have both mixed-case characters and spaces in member names, but case and spaces are ignored when member names are stored. This means that `North America`, `NorthAmerica`, and `north america` are equivalent member names in a database. If you dropped the following file onto the dimension editor, you see the member `North America` in your hierarchy, as that is the first occurrence that is encountered during file processing.



	A
1	North America
2	NorthAmerica
3	north america

File size

No file size limit applies when you drop text files into the dimension editor, but large files might cause your web browser to timeout while you wait for a response. The text file is still processed and the members are inserted into the dimension hierarchy.

Weights

The weight property controls how a child member rolls up to its immediate parent, whether that child is also a parent of another consolidation or a leaf member. Top level consolidations do not have a weighting.

Weight only applies to numeric members. The default weight for a member is 1, this gives a member a positive value. Weight can be used to change a positive value to a negative value, often -1. For example, if the unit price for a product is EUR 50 and the discount is EUR 5, apply a weight of -1 to the discount member. Applying a weight keeps the addition result logical.

You can also weight members as zero to keep them out of a total. For example, say you have a consolidation member called Drivers, and this is made up of Floorspace, Headcount, and Population growth. You would weight floorspace, Headcount, and Population growth as zero because it makes no sense to add them together.

Gross margin example

You have an Accounts dimension with the following members: Gross margin, Sales, and Variable costs. The Gross margin member is calculated as Sales - Variable costs. In the dimension, it would be structured like this:

- Gross profit
  - Sales
  - Variable costs

You would give Sales a weight of 1 and Variable costs a weight of -1.

Format dimension members

Modelers can set the format for values in dimension members in the dimension editor. The default format is General.
For example, you can format a member as a date picker, as a currency value, or as scientific notation. You can also create custom formats.

Note: You can also change the format of values directly in a view or cell view, see "Change the format of data in a view" on page 58.

Procedure

1. In the **Data** tree, click **Dimensions**, right-click the dimension that you want to edit and click **Edit dimension**.
2. Right-click the dimension member, or members and click **Set format**.
3. Select the format.
   For example, for a date format, select **Date**, and then click **OK**.

   If you selected custom format, first type `c:` then, type the format in the field. See "Custom character formats" on page 195.

4. To display the format attributes in the dimension editor, click 🏷
5. You can set the format for all members in the dimension at the same time in the **Format** attributes column. Right-click the **Format** attributes header and select **Set format**.

What to do next

To format a member as text, set the type to string, see “Change the type of members to numeric or text” on page 194.

Member formats

You can format dimension members in the dimension editor.

You can choose to set the format for the value of a member as one of the following formats.

Note: The examples that are shown use the actual value of -1234.567. The number changes based on the format used.

General

Displays numbers without commas to separate digits to the left of the decimal point. Negative values are prefixed with a minus sign (-).

`-1234.57`

Fixed

Displays numbers without commas to separate digits to the left of the decimal point. Negative values are surrounded by parentheses.

`(1234.57)`

Comma

Commas separate every third digit to the left of the decimal point.

`(1,234.57)`

Rounded

Commas separate every third digit to the left of the decimal point, rounded up to the nearest whole number. You can see the actual value by right-clicking in the cell.

`(12,346)`

Percentage

Multiplies numbers by 100 and displays a following percent sign (%). Digits to the left of the decimal point do not use commas, and negative values are prefixed with a minus sign (-).

`-123456.70%`

Scientific

Displays numbers in scientific notation. Scientific notation is a way of expressing large or small numbers. For example, the number 123,000,000,000 can be written as 1.23E+11.

Negative values are prefixed with a minus sign (-).

`-1.2E+3`

Accounting

Displays numbers with currency symbols and decimal points in a column. Negative values are surrounded by parentheses.

`$(1,234.57)`

Currency

Displays numbers with the currency symbol that is specified for your computer. Commas separate every third digit to the left of the decimal point. Negative values are surrounded by parentheses.

`($1,234.57)`

Currency (rounded)

Displays numbers with the currency symbol that is specified for your computer. Commas separate every third digit to the left of the decimal point, rounded up to the nearest whole number. You can see the actual value by right-clicking in the cell. Negative values are surrounded by parentheses.

`($1,235)`

Date

Gives you a date picker, and displays dates in a predefined format: `mm/dd/yyyy`.

`01/23/1989`

Time

Displays time in a predefined format: `hh:mm:ss`.

`12:30:00`

You can also define custom formats.

Format examples

The following list shows the standard formats that are used in the dimension editor.

Note: d: means that ICU formatting is used. ICU is the standard format for IBM Planning Analytics Workspace.

Fixed

`d:0.00;(0.00)`

Comma

`d:#,##0.00;(#,##0.00)`

Percentage

`d:#.00%`

Scientific

`d:0.0E+0`

Accounting

`d:'$'#,##0.00;'$'(#,###.00)`

Currency

`d:'$'#,##0.00;('$'#,###.00)`

Currency (rounded)

`d:'$'#,##0;('$'#,###)`

Date

`c:MM/dd/yyyy`

Time

`d:H:mm:ssICD`

You can create custom formats by modifying these examples. To find out more about the syntax, see "Custom character formats" on page 195, "Numeric values" on page 198, and "Date and time display formats" on page 197.

Change the type of members to numeric or text

You can change the properties of a member to make it a numeric value or a string (text), and you can change the weighting of a member.

Make sure that all members that are strings in a cube are in the same dimension, along with any pick list members. The dimension that contains string members and pick list members must be the last dimension in the cube.

Procedure

1. To open the **Member Properties** pane, select a member in the dimension editor and click ⓘ.
2. In the **Member Properties** pane, you can change the following member properties:

   Type

   You can set a member type to Numeric or String. You can change only the member type of leaf items.

Note: Dimensions that contain members that have the String type, or are pick lists, must be last in a cube. String type members and pick list members must be in the same dimension.

Weight

Weight can be used to change a positive value to a negative value, often -1.

For example, if the unit price for a product is EUR 50 and the discount is EUR 5, apply a weight of -1 to the discount member. Applying a weight keeps the addition result logical. To find out more, see “Weights” on page 192.

Note: To toggle to the **Member Properties** pane, click ⚙.

Custom character formats

You can create a custom format in the dimension editor to display characters. This information applies to custom formats for dimension members in the dimension editor only.

To display text, change the member properties to a String type.

1. In the dimension editor, right-click the dimension member that you want to set the format for and select **Set format** > **Custom format**.
2. Type your custom format.

In IBM Planning Analytics Workspace, you must start the custom format with c:

Format expressions for characters can have one section, or two sections separated by a semicolon (;). If you use one section, the format applies to all string data that can occur in the cell. If you use two sections, the first section applies to string data, and the second section applies to null values and zero-length strings.

For example:

```
c:<@@@;"No Value"
```

This format displays three lower-case characters if the cell contains string data, or the string No Value if the cell contains a null value or a zero-length string.

Note: Dimensions that contain members are formatted as strings, or are pick list, must be last in a cube. String type members and pick list members must be in the same dimension.

The following table describes how to construct a format string for a string element:

Format String Character	Description
@	Character placeholder. If the string has a character in the position where the at symbol (@) appears, that character displays. If no character appears in that location, a space displays. **Example:** Suppose a cell contains the following string: The quick brown fox If you apply this format string: `c:@@@@@` Displays: n fox **Note:** Placeholders are populated from right to left unless you enter an exclamation point (!) character in the format string.

Format String Character	Description
&	Character placeholder. If the string has a character in the position where the ampersand symbol (&) appears, that character displays. In this case, a space is considered a character and will be displayed. If no character appears in that location, nothing displays. **Example:** Suppose a cell contains the following string: The quick brown fox If you apply this format string: `c:&&&&&` Displays: nfox **Note:** Placeholders are populated from right to left unless you enter an exclamation point (!) character in the format string.
<	Displays all characters in lowercase. < must be used in conjunction with either @ or &, for example: `c:<&&&&&`
>	Displays all characters in uppercase. > must be used in conjunction with either @ or &. **Example:** Suppose a cell contains the following string: The quick brown fox If you apply this format string: `c:>@@@@@` Displays: N FOX
!	Forces placeholders to fill from left to right. ! must be used in conjunction with either @ &. **Example:** Suppose a cell contains the following string: The quick brown fox If you apply this format string: `c:!>@@@@@` Displays: THE QU

Date and time display formats

The following table lists characters that can appear in a format string for date and time formats.

Format String Character	Description
:	Time separator. (In some locales, other characters may be used to represent the time separator.) This character separates hours, minutes and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.
/	Date separator. (In some locales, other characters may be used to represent the date separator.) The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.
C	Displays the date as ddddd and displays the time as ttttt, in that order. Displays only date information if there is no fractional part to the date serial number. Displays only time information if there is no integer portion. Example: 10/10/98 05:12:12
d	Displays the day as a number without a leading zero (1-31).
dd	Displays the day as a number with a leading zero (01-31).
ddd	Displays the day as an abbreviation (Sun-Sat).
dddd	Displays the day as a full name (Sunday-Saturday).
ddddd	Displays the date as a complete date (including day, month, and year), formatted according to the long date setting recognized by your system. For Microsoft Windows, the default long date is m/d/yy.
dddddd	Displays a date serial number as a complete date (including day, month, and year), formatted according to the long date setting recognized by your system. For Microsoft Windows, the default long date format is mmmm dd, yyyy.
w	Displays the day of the week as a number. (1 for Sunday through 7 for Saturday).
ww	Displays the weeks of the year as a number (1 - 54)
m	Displays the month as a number without a leading zero (1 - 12). If m immediately follows h or hh, the minute rather than the month displays.
mm	Displays the month as a number with a leading zero (01 - 12). If m immediately follows h or hh, the minute rather than the month displays.
mmm	Displays the month as an abbreviation (Jan - Dec).
mmmm	Displays the month as a full month name (January - December)

Format String Character	Description
q	Displays the quarter of the year as a number (1 - 4).
y	Displays the day of the year as a number (1 - 366).
yy	Displays the year as a two-digit number (00 - 99).
yyy	Displays the year as a four-digit number (0100 - 9999).
h	Displays the hour as a number without leading zeros (0 - 23).
hh	Displays the hour as a number with leading zeros (01 - 23).
n	Displays the minute as a number without leading zeros (0 - 59).
nn	Displays the minute as a number with leading zeros (00 - 59).
s	Displays the second as a number without leading zeros (0 - 59).
ss	Displays the second as a number with leading zeros (00 - 59).
t t t t t	Displays a time as a complete time (including hour, minute, and second), formatted using the system time separator. A leading zero displays if the time is before 10:00 AM or 10:00 PM. For Microsoft Windows, the default time format is hh:mm:ss.
AM/PM	Uses the 12-hour clock. Displays an uppercase AM with any hour before noon; displays an uppercase PM with any hour between noon and 11:59 P.M.
am/pm	Uses the 12-hour clock. Displays a lowercase AM with any hour before noon; displays a lowercase PM with any hour between noon and 11:59 P.M.
A/P	Uses the 12-hour clock. Displays an uppercase A with any hour before noon; displays an uppercase P with any hour between noon and 11:59 P.M.
a/p	Uses the 12-hour clock. Displays a lowercase a with any hour before noon; displays a lowercase p with any hour between noon and 11:59 P.M.
AMPM	Uses the 12-hour clock. Displays the AM string literal with any hour before noon; displays the PM string literal with any hour between noon and 11:59 P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. For Microsoft Windows, the default format is AM/PM.

Numeric values

Format expressions for numbers have up to four sections separated by semicolons (;). The number of sections determines which types of values are affected.

- If a format has one section, that section applies to all values.
- If a format has two sections, the first section applies to positive values and zeros, and the second section applies to negative values.
- If a format has three sections, the first section applies to positive values, the second section applies to negative values, and the third applies to zeros.

- If a format has four sections, the first section applies to positive values, the second section applies to negative values, the third applies to zeros, and the fourth applies to NULL values.

The following table describes how to construct a format string for numeric values:

Format String Character	Description
# (number sign)	Digit placeholder. The # placeholder displays only significant digits and does not display insignificant zeros. In the decimal value .90, the 0 is considered insignificant. The value would be displayed as .9 when the # placeholder is used. If a number has more digits to the right of the decimal point than there are placeholders in the format string, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed.
	The following examples illustrate the use of the # placeholder. **Example** Value: 123.896 Format String: #.## Displays: 123.9 **Example** Value: 456.873 Format String: #.## Displays: 456.87 **Example** Value: 34.5678 Format String: #.### Displays: 34.568 You can combine the # and 0 placeholders in a format string.

Format String Character	Description
0 (zero)	Digit placeholder. The 0 placeholder displays insignificant zeros if a number has fewer digits than there are zeros in the format string. If a number has more digits to the right of the decimal point than there are placeholders in the format string, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. The following examples illustrate the use of the 0 placeholder. **Example** Value: 23.896 Format String: 0.00 Displays: 23.90 **Example** Value: 16.8 Format String: 0.000 Displays: 16.800 **Example** Value: 7.12 Format String: 000.0 Displays: 007.1 You can combine the # and 0 placeholders in a format string.
E- E+ **e- e+**	Scientific format. If the format string contains at least one digit placeholder (0 or #) to the right of E-, E+, e-, e+, the number displays in scientific format and E or e is placed between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents.
- + $ ()	Displays a literal character. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks. (" "). Numeric Value: -1000.00 Format String: `($-#.##)` Displays: ($-1000.00) The following characters cannot be displayed as literals: a, c, d, h, m, n, p, q, s, t, w, y, /, :, #, 0, %, E, e, comma(,), period(.), @, &, <, >, and !

Format String Character	Description
\	Displays the next character in the format string. Numeric Value: 100 Format String: `\t\o\t\a\l\=#` Displays: total=100
"ABC"	Displays the string inside the double quotes. (In this example, ABC would display.) Numeric Value: 100 Format String: `#" units"` Displays: 100 units

Add members to a dimension
You can add members to a dimension one at a time or by copying and pasting a number of members from an editor such as Microsoft Excel.

You can also import dimension members from a file, see Import members and attributes into a dimension.

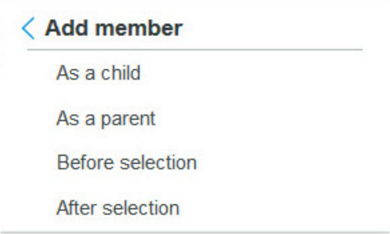
Procedure

1. In the Data tree, go to the database that contains the dimension that you want to edit.
2. Click **Dimensions**, right-click the dimension that you want to edit and click **Edit dimension**.
3. To add a member, click ⊕ type a name and select the member type, either:
   - **Numeric**
   - **Consolidated**
   - **String**. This type cannot be used if the member is a child of a consolidated member.

   If you add a member as a parent, the member is always added as a consolidated member.

   If you want to add another member at the same level, type the new member name and press Enter.

   Click **Commit** when you finish entering members at that level, or press CTRL + Enter.
4. To add more members at parent or child level, right-click an existing member and add a member as a parent, child, or at the same level before or after the selection.

   Tip: You can also add a member into a dimension at the same time by copying and pasting the list of members into the **Dimension Editor** from an editor, or from Microsoft Excel.
5. You can choose what you see in the dimension editor by right-clicking an existing member and choose an option to keep, hide, move, or delete members.

   If you select **Keep**, the other members are hidden. To narrow the selection down to show parents or children of selected members in a dimension, select and right-click the members and click **Show members** > **Parent** or **Children**. You can show all members by right-clicking a member and selecting **Unhide all**.

What to do next

- Format the members. For example, you can format a member as a date or time picker, as a currency value, or as scientific notation. See "Member formats" on page 193.
- Change the weighting of a member. See "Change the type of members to numeric or text" on page 194.
- Create attributes for dimension members by clicking 🏷. Attributes provide extra information about dimension members, see "Member attributes" on page 209.
- Drag a dimension member as parent, child, or peer of other members. See "Drag members as children or peers " on page 202.

Drag members as children or peers

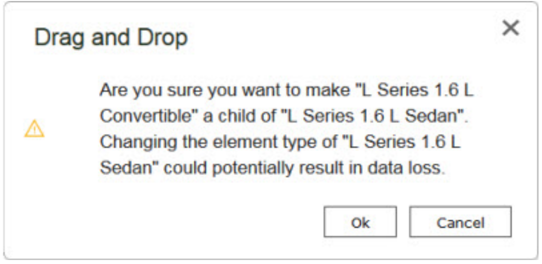
If you are in a **Dimension Editor**, you can drag a member as a child of another member or as a peer of another member.

About this task

If you are sorting the members of your dimension in the **Dimension Editor**, the members are dropped in to the dimension according to the sort order.

The following limitations apply:

- You cannot drag a member as a parent of another member.
- You can multi-select members but the member that is dragged is always the last selected member. You cannot drag multiple members.
- If you drag a member as a child of another member, a dialog box warns you about potential data loss. This warning appears only if the dimension that you are editing is part of a cube and this action converts a leaf level member to a consolidated member.



Drag and Drop

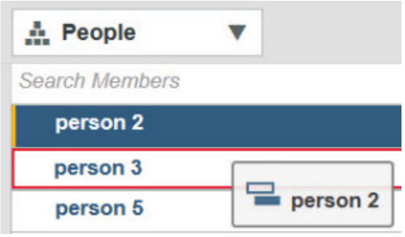
⚠ Are you sure you want to make "L Series 1.6 L Convertible" a child of "L Series 1.6 L Sedan". Changing the element type of "L Series 1.6 L Sedan" could potentially result in data loss.

Procedure

1. In the **Data** tree, go to the database that contains the data that you want to view.
2. Right-click an existing dimension, and click **Edit dimension**.
3. In the dimension editor, select the member that you want to drag.

   When you start to drag the selected member, a tooltip text appears. The icon inside the tooltip changes based on how you want to insert the member. As you drag the member that you want to move, a red line or border appears that indicates where you can drop the member.
4. Drag the member onto another member to make it a child of that member.

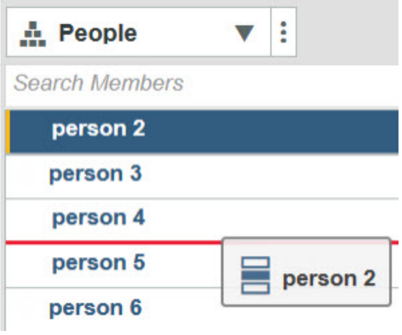
   When you drag a member onto another member, the parent member has a red border that indicates a parent-child relationship.



Note: When you drop a member as a child, the parent member becomes a consolidated member. When you remove the child from the parent, the parent might become a childless consolidated member. The +/- icon appears for all consolidated members even if they have no children.

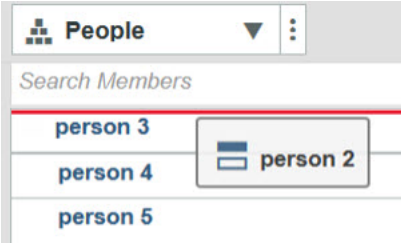
5. Drag the member between two members to drop it as a peer of another member.

   When you drag a member between two existing members, a red line appears between the two members to indicate where you can drop the member.

6. Drag the member before or after an existing member.

   When you drag a member before an existing member, a red line appears before the member to indicate that you can drop before this member.

   When you drag a member after an existing member, a red line appears after the member to indicate that you can drop after this member.

Time dimensions

A time dimension defines the time periods that are used in your model. Most models use a time dimension.

Time dimensions can specify financial accounting periods, or the dates of sales transactions.

You can create a time dimension by using the **Customize as Time** option when you create a new dimension. You can also create a time dimension in the same way as you would any dimension, either by importing the members, or by adding them manually.

If you create a time dimension by using the **Customize as Time** option, you can select the time range that you want to create. You can also select the time periods that you want to create, from Years, Quarters, and Months.

A time dimension that is created with the **Customize as Time** option has the following member attributes:

Last Period
The final time period in the sequence.

First Period
The initial time period in the sequence relative to the current time.

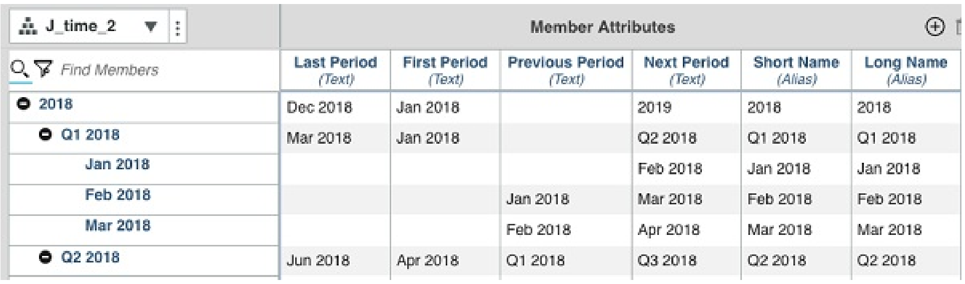
Previous Period
The previous time period in the sequence.

Next Period
The next time period in the sequence.

Short Name
An alias. The short name is populated with the Member ID (the member name) - for example Jan 2018. You can modify the short name by typing in the cell.

Long Name
An alias. The long name is populated with the Member ID. You can modify the long name by typing in the cell.

To view the time attributes, open the time dimension in the dimension editor and click 🏷.

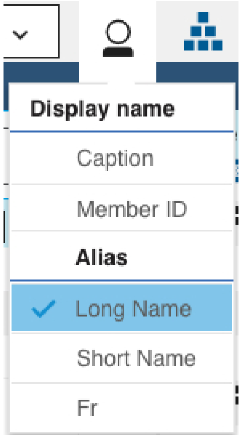


	Last Period (Text)	First Period (Text)	Previous Period (Text)	Next Period (Text)	Short Name (Alias)	Long Name (Alias)
2018	Dec 2018	Jan 2018		2019	2018	2018
Q1 2018	Mar 2018	Jan 2018		Q2 2018	Q1 2018	Q1 2018
Jan 2018				Feb 2018	Jan 2018	Jan 2018
Feb 2018			Jan 2018	Mar 2018	Feb 2018	Feb 2018
Mar 2018			Feb 2018	Apr 2018	Mar 2018	Mar 2018
Q2 2018	Jun 2018	Apr 2018	Q1 2018	Q3 2018	Q2 2018	Q2 2018

You can add attributes, for example, you might want to add translations for the text. To find out more, see "Create member attributes" on page 210.

You can use time period attributes in calculations.

Tip: You can choose to display the **Short Name** or **Long Name** attributes in cube views instead of the Member ID or the Caption. The Member ID is the name of the member, and the Caption is the default display name. In the cube view, click the time dimension tile, click ✎ to open the set editor, click 👤, and select the display name type.



Create a time dimension

You can create a time dimension by creating a new dimension and selecting the **Customize as Time** option.

The **Customize as Time** option guides you through the steps to create a basic time dimension.

Note: Customize as Time is only available for a dimension with one hierarchy and no members.

Procedure

1. In the **Data** tree, go to the database where you want to create the dimension.
2. Right-click **Dimensions**, and click **Create dimension**.
3. Click **Customize as Time**.
4. Select the year that you want the dimension to start from in **Start Year**.
5. Select the year that you want the dimension to finish on in **End Year**.
6. Select the granularity of the dimension.

Months
This option gives you three levels: Months, Quarters, Years.

Quarters
This option gives you two levels: Quarters, Years.

Years
This option gives you one level: Years.

7. Check the preview to see whether it matches what you expect, then click **Create**.

   You can modify the dimension in the dimension editor after you created it.

Reserve a dimension

When you work in the dimension editor, any changes you make are instantly applied so that anyone who has access to the database can see the dimension. You can reserve a dimension to prevent other people from editing the dimension before your changes are complete. Reserving a dimension means that any changes that you make are not saved until you commit the changes.

Procedure

1. In the dimension editor, click ⋮ next to the dimension tile, and select **Reserve**. This icon indicates that the dimension is reserved and you can edit it: 👤 ⋮

   You can edit only the dimension members and hierarchy when a dimension is reserved, you can’t edit attributes or member security.

   Another person trying to edit a reserved dimension sees this icon next to the dimension tile: 🔒 ⋮.

   They can release a reserved dimension by clicking ⋮ and selecting **Release**.

2. To undo changes to the dimension before the data is committed back to the database, click ⋮ and select **Cancel**.

3. When your changes are complete, click ⋮ and select **Commit**.

Find and filter members

Modelers can find or filter on members in the dimension editor.

In addition to filtering by member name, you can filter by sets.

Procedure

1. Right-click a dimension, and click **Edit dimension** or **Edit dimension in new tab**.

2. Type a phrase in the **Find Members** field and click ⏷ to filter on the search phrase.

   Filtering narrows down the list of members to members that contain a search phrase.

3. Type a phrase in the **Find Members** field and click 🔍 to highlight all instances of the phrase in the dimension.

   The first member in the dimension that contains the phrase is highlighted. The total number of members that are found is displayed, and you can cycle through the members by clicking the up and down arrows 1 of 10 ⌃ ⌄ in the **Find Members** field.

Tip: You can switch between Filter and Find by clicking the find 🔍 or filter ▽ icons.

4. To filter on the dimension editor by set, click ⋮ and select **Filter by set**. A list of the available sets appears. Filter many sets by typing a search phrase in the box at the top of the list of sets.

   To remove the set, click ⋮, and select **Clear set filter**.

   Note: When you filter by set, you cannot edit the hierarchy structure. Filter by set is used to make editing hierarchy attributes and member security easier.

5. To edit a set from the dimension editor, click ⋮ ✎ .

Edit a dimension

You can edit dimensions with the **Dimension Editor** when you are a Modeler or an Administrator.

Note: All changes to your dimension are instantly applied to the database.

Procedure

1. In edit mode, in the **Data** tree, go to the database that contains the data that you want to view.
2. Right-click a dimension, and click **Edit dimension**.
3. To create another dimension, click ⋮ next to the dimension tile and click **Create New Dimension**.
4. To create another hierarchy, click ⋮ next to the hierarchy tile and click **Create New Hierarchy**.

What to do next

For more information about creating hierarchies, see “Hierarchies” on page 208.

Dimension settings

Modelers can view and edit dimension security settings, properties, and attributes in the settings editor.

Note: Some dimension properties cannot be edited directly. They are updated by the system.

Procedure

1. Expand the database that you want to manage dimension settings for in the **Data** tree, and right-click **Dimensions**.
2. Click **Edit settings**.
3. Click 🏷 to view dimension attributes, click 🔒 to view dimension security settings, and click ☰ to view dimension properties.

What to do next

To find out more about dimension security settings, see “Secure cubes, dimensions, processes, and chores” on page 243.

Dimension properties
Modelers can view and edit dimension properties in the settings editor. Some of these properties cannot be edited in the settings editor, they are updated by the system.

Member	Description
REPSTATUS	If a dimension is replicated on the server, the value of REPSTATUS is Copied. Otherwise the REPSTATUS value is empty.
SYNCSUBSETS	For replicated cubes, this property indicates whether subsets are synchronized when the associated dimension is synchronized.
SYNCATTRIBUTES	For replicated cubes, this property indicates whether element attributes are synchronized when the associated dimension is synchronized.
LAST_TIME_UPDATED	Time (GMT) at which a dimension was last updated. The time format is yyyymmddhhss.
SORTCOMPONENTSSENSE	When the immediate components (children) of a consolidation are sorted, this property stores the sense applied to the sorting. Components can be sorted in either the ASCENDING or DESCENDING sense.
SORTELEMENTSSENSE	When dimension elements are sorted automatically, this property stores the sense applied to the sorting. Elements can be sorted in either the ASCENDING or DESCENDING sense.
SORTCOMPONENTSTYPE	This property indicates the type of sorting that is applied to the immediate components (children) of a consolidation. There are two types of sorting: BYNAME and BYINPUT. The BYINPUT sort does not actually enforce any sorting, it leaves the components in the order they appeared the last time the dimension was saved.
SORTELEMENTSTYPE	When dimension elements are sorted automatically, this property stores the type of sorting used. There are four types of element sorting: BYNAME, BYLEVEL, BYHIERARCHY, and BYINPUT. The BYINPUT sort does not actually enforce any sorting, it leaves the elements in the order they appeared the last time the dimension was saved. If left blank, BYINPUT applies. SORTELEMENTSTYPE applies to all dimension elements, both consolidations and leaf elements.
DEMANDLOAD	Not currently used.
DEFAULT_HIERARCHY	Not currently used.
LOCK	Indicated when a dimension is locked.
REPSRCNAME	For replicated dimensions, this property stores the name of the source dimension.
REPLICATION	For replicated cubes, this property stores the name of the replication connection with which the replicated cube is associated.

Member	Description
LASTREPSYNCCHANGEDTIME	
USESEPARATECONTROLCUBES	Not currently used.
VISIBILITY	
ALLLEAVESHIERARCHYNAME	This property stores the name given to the all leaves hierarchy. The all leaves hierarchy is present in any dimension that contains more than one hierarchy.

Dimension attributes
Dimension attributes provide information about the dimension. You can view and edit the values for dimension attributes, but you cannot add them in the settings editor. Modelers can add dimension attributes by creating TurboIntegrator processes, or by creating them directly in IBM Cognos® TM1 Architect.

For example, dimension attributes can be used to display the language name in different languages by using an alias.

Typical examples of dimension attributes are:

CAPTION
The caption attribute is used to display dimension names in other languages.

DIMENSION_TYPE
Typical values are: CALCULATION, METRIC, VERSIONS, TIME, GENERIC.

SWITCHOVER
This is a date value that defines the switchover period for a version dimension.

Hierarchies

You can use hierarchies in dimensions to see alternative rollups of the data in the same view. For example, you might have a hierarchy that shows an organization that is rolled up by region, and a hierarchy that shows an organization that is rolled up by vice-president.

You can also create hierarchies of member attributes such as color, size, and type. Use these hierarchies for filtering and analysis.

Use the hierarchy menu in the **Dimension Editor** to create and maintain hierarchies.

You can view this video to see an overview of hierarchies:

All dimensions include a single hierarchy by default.

Here are some of the benefits of using hierarchies:

- Hierarchies can improve query performance.
- You can turn attributes into hierarchies. Modeling attributes as hierarchies instead of dimensions can save memory space. If you use hierarchies, you can have cubes with fewer dimensions. Hierarchies act as virtual dimensions.
- Hierarchies give you greater flexibility. A single dimension can contain multiple hierarchies; hierarchies act as virtual dimensions, and you can display multiple hierarchies in the same view.

Note: Creating multiple hierarchies in a dimension is disabled in Planning Analytics Workspace by default. You must set the `EnableNewHierarchyCreation` configuration parameter in the `tm1s.cfg` file to `true` to enable hierarchy creation in Planning Analytics Workspace. For more information, see Parameters in the tm1s.cfg File in the *Planning Analytics Installation and Configuration* documentation.

Sort order

The sort order for hierarchies is determined on the following basis:

- The default hierarchy (typically, the hierarchy with the same name as the dimension) appears first.
- The all-leaves hierarchy (typically called Leaves) appears last.
- All other hierarchies are sorted alphabetically by their caption.

Create a hierarchy

You can create a hierarchy with the **Dimension Editor** when you create a dimension or in an existing dimension.

You can also create a hierarchy by saving an existing hierarchy with a new name, and you can create a hierarchy from an attribute.

Procedure

1. To create a hierarchy from within the dimension editor, click ⋮ next to the hierarchy tile and then click **Create hierarchy**.
2. To create a hierarchy from the Data tree, do the following steps:
   a) Right-click a dimension and click **Create hierarchy**.
   b) Enter the name of the new hierarchy and click **Create**.
   c) Add members to the hierarchy by clicking ⊕, or by dragging a formatted file onto the empty hierarchy.
3. To create a hierarchy from an existing hierarchy, do the following steps:
   a) Click ⋮ next to the hierarchy tile and select **Save hierarchy as**.
   b) Name the hierarchy and click **Save**.

      The members and attributes of the original hierarchy are copied.
4. To create a hierarchy from an attribute, right-click an attribute in a dimension and click **Create hierarchy**.

   For more information, see "Member attributes" on page 209.
5. To edit an existing hierarchy from the Data tree, in the content tree, right-click a hierarchy and click **Edit hierarchy**.
6. To delete a hierarchy, in the Data tree, right-click a hierarchy and click **Delete hierarchy**.

Member attributes

Member attributes help to explain or describe a member in a dimension.

For example, suppose that you have a dimension that lists cars. You could have attributes that describe fuel type, engine size or type, and number of doors.

You can show attributes in a table and you can select which attributes you want to show. You can sort on attributes, and you can convert attributes into a hierarchy. Converting attributes into hierarchies is useful if you want to group dimension members by attribute. For example, suppose that you want to display cars that have a gasoline engine, you can add an attribute called Engine type, then create a hierarchy from the Engine type attribute. You would then be able to group members by Engine type attribute in the view.

You add member attributes in the dimension editor.

If you want to show only members that have specific attributes, create a set by filtering on attributes. To find out more, see "Search for members in a set" on page 146.

Create member attributes

Creating attributes for dimension members helps users to understand the information in reports and views. If you are logged in with the modeler role, you can create attributes in the dimension editor. You can create hierarchies from attributes.

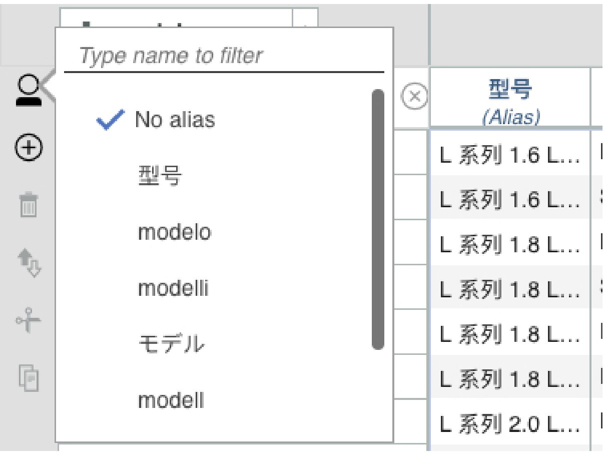
Procedure

1. Make sure that you are in edit mode.
2. Right-click the dimension, and click **Edit dimension**.
3. Click 🏷 to open the attributes pane.
4. Select a member and click ⊕ in the **Member Attributes** bar.
5. Enter the attribute name, and select the attribute type.

   A column is created for each attribute, with the attribute type in the header.

   You can use the Alias attribute type to specify alternative names for dimension members.

   Note: Hide the attribute type by clicking ⋮ and selecting **Hide attribute type**.
6. To view an alias in the dimension editor instead of the dimension name, click 👤 and select the alias. To clear the alias and view the dimension member name, select **No alias**.



7. When you have created your attribute, you can type or paste the attribute values into the cells. To paste values copied from a text file or spreadsheet, press CTRL+ V or CMD+ V.

   To learn more, see "Tutorial: Creating dimensions and hierarchies" on page 356.

Captions for members

A caption is an attribute that has the alias attribute type, and a name of `Caption`. You can choose whether to display the member caption, the member ID, or an alias in the set editor. If a caption exists, the caption is the default display name.

Captions can also be created for other objects such as cubes, and dimensions.

Caption

In IBM Planning Analytics Workspace, the value for caption must be unique. The caption is the default value.

The caption can be overwritten by localized captions. Localized captions can be created with a TurboIntegrator process, see "Create and edit processes" on page 248, and "Translate members" on page 263.

Member ID

The member ID is the name that a member was given when it was first created. The member ID cannot change.

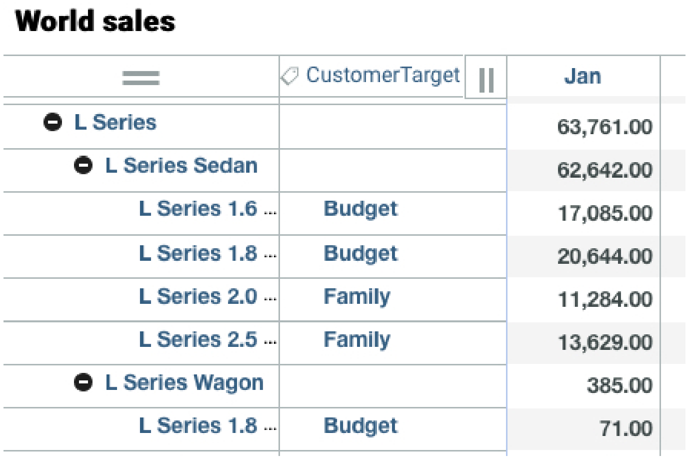
Aliases

Aliases are attributes that have the attribute type of alias. Captions must have the alias type in Planning Analytics Workspace.

Show member attributes in the cube view

Some dimension members have attributes. Attributes help to explain or describe a member in a dimension. You can show member attributes in a table.

For example, suppose you have a dimension with car models with an attribute called CustomerTarget, and you want to see the customer target for each car model in the table. You can choose to show or hide this attribute.



World sales

═	CustomerTarget ‖	Jan
⊖ L Series		63,761.00
⊖ L Series Sedan		62,642.00
L Series 1.6 ...	Budget	17,085.00
L Series 1.8 ...	Budget	20,644.00
L Series 2.0 ...	Family	11,284.00
L Series 2.5 ...	Family	13,629.00
⊖ L Series Wagon		385.00
L Series 1.8 ...	Budget	71.00

Procedure

1. Right-click either the row selector ═, or the column selector ‖, and select **Show attributes**.
2. Select the attribute that you want to show in the **Available attribute** pane, and click → to move the attribute into the **Selected attribute** pane.

   You can select several attributes and you can choose the order in which the attributes are displayed by moving them using the up and down arrows.
3. Click **OK** to save your choices and to return to the table.
4. You can sort the attributes in either ascending or descending order. Right-click either the row selector, or the column selector, and select either **Sort ascending** or **Sort descending**.
5. To hide attributes, right-click the row or column selector, and select **Hide attribute**.

Edit member attributes

You can edit member attributes in the dimension editor when you create or edit members.

Procedure

1. To open the attributes pane, in the dimension editor, click 🏷.
2. To edit an attribute, right-click the title of the attribute.

You can add, delete, keep, hide, show, sort, or search for attributes.

3. To change the attribute cell values, type in the cell.

   You can also right-click a cell value, and cut or copy a cell value. To paste a value, press CTRL+ V or for Microsoft Windows computers, or CMD+ V for Apple Macintosh computers. You can paste a single value across multiple selected cells.

What to do next

For more information, see "Data entry" on page 117.

Pick lists

A pick list contains values that a user can select in a cell.

The values in a pick list correspond to the members of a dimension or set of a dimension. If the members of the dimension or the set change, the values available in the pick list also change. A pick list can also contain a static list of values that you specify when you create the pick list.

You can create the following kinds of pick list:

Static
The pick list only contains the values that you specify.

Subset
The pick list references the values in the set. If the values in the set change, then the values in the pick list also change.

Dimension
The pick list references the values in the dimension. If the values in the dimension change, then the values in the pick list also change.

You can also create pick lists using control cubes, see "Create a pick list control cube" on page 213.

A pick list is useful when staff managers do performance planning; they can assign to their staff a performance grade by using pick lists to select from a fixed list of **Low**, **Medium**, **High**, and **Excellent**, instead of typing a free form text string. If you set a pick list on a member with an existing format, the operation removes the format.

A pick list could also be used to select job codes, or to select months.

Important considerations

Cell edits applied through data spreading operations and TurboIntegrator processes are **not** validated. Edits applied through either of these methods can result in cell values that do not conform to valid pick list values. Data spreading can be applied to cells containing pick lists only through the data spreading dialog boxes; data spreading shortcuts cannot be used in cells containing pick lists.

When defining a pick list that contains numeric values, you must use the Cultural Invariant style, which uses a period (.) as a decimal separator. The Cultural Invariant style is equivalent to English style.

Create a pick list

Create a pick list either by referencing a set or a dimension in a member attribute that is called Picklist, or by adding static values to the Picklist attribute.

A pick list that references a set or a dimension contains all the values that are in the set or dimension. If the members change, the values in the pick list also change. A pick list that contains static values contains only those values.

Note: A dimension that contains pick lists must be last in the cube. String type members and pick list members must be in the same dimension in a cube.

Procedure

1. Open the dimension that references the pick list in the dimension editor.

2. Click 🏷 to open the **Attributes** pane, click ⊕, and add a text attribute called `Picklist`.
3. To reference a set, add the following Picklist attribute value:

   ```
   Subset:dimensionname:setname
   ```

   See the following example:

   ```
   Subset:Years:months
   ```

4. To reference a dimension, add the following Picklist attribute value:

   ```
   Dimension:dimensionname
   ```

   For example:

   ```
   Dimension:Employees
   ```

5. To reference the members of a named hierarchy within a dimension, add the following Picklist attribute value:

   ```
   dimension:dimensionname\:hierarchyname
   ```

   See the following example:

   ```
   dimension:Products\:Europe
   ```

6. To create a static pick list, add the following Picklist attribute value:

   ```
   static:value1:value2:value3:value4
   ```

   See the following example:

   ```
   static:none:January:February:March:April:May:June:July:August:September:October:November:Dece
   mber
   ```

   To include a null value at the beginning or in the middle of a static pick list, use two consecutive colons in the pick list definition. For example, the code: `static:value1:value2::value3:value4` results in a pick list with a null value between `value2` and `value3`.

   To include a null value at the end of a static pick list, insert a colon without a following value at the end of the pick list definition. For example, `static:value1:value2::value3:value4:` results in a pick list with a null value that follows `value4`.

7. Select the picklist dimension member and open the **Member Properties** pane by clicking ⊕, then click ◉to set the member properties. Set the type to **String**.

What to do next

To test that the pick list is successfully created, add a view with the dimension that contains the pick list.

Create a pick list control cube

Use a pick list control cube to define which individual cells a pick list is available from.

About this task

Using rules, you can define pick lists for any section of a cube, from a single cell to the entire cube.

Important: To create pick list control cubes, you must use IBM Planning Analytics version 2.0.8 and later.

A pick list control cube contains the same dimensions as the cube it was created from, with an extra dimension named **}Picklist**. **}Picklist** has one member that is named Value.

Procedure

1. Right-click the cube that you want to create the pick list control cube for, then click **Edit pick list cube**.

   The rule editor and a view of the new control cube is opened. You can define the pick list either in the view, or in the rule editor.
2. To define the pick list in the view do the following steps:
   a) Change the view of the control cube so that you see the cells that you want to define pick lists for.
   b) Type a pick list definition in the cell.

      You can enter any of the pick list types in the control cube: static, set, or dimension. See "Create a pick list" on page 212 for examples.
3. To define the pick list in a rule, do the following steps:
   a) In the rule editor that was created, use a standard rules area definition to specify the cells that you want the pick list to appear in.
   b) Immediately after the area definition, type `=S:`. S is the string qualifier, indicating that the rule applies to string cells.
   c) Immediately after the string qualifier, type a pick list definition, which is enclosed in single quotation marks, then enclosed in parentheses. For example, `('static:spring:summer:winter:fall')`
   d) Immediately after the pick list definition, type a semi-colon (;) to end the rule statement.
4. To prevent a cell from displaying a pick list, type none in the pick list control cube cell or use `(’none’)` as the formula in a rules statement. For example, `[’season’]=S:(’none’);`.

Pick list rule examples

Rule statement	Description
`['size','shirts']=S:('static:16:17:18');`	This rule statement indicates that any cell identified by the members **size** and **shirts** displays a static pick list with the values 16, 17, and 18.

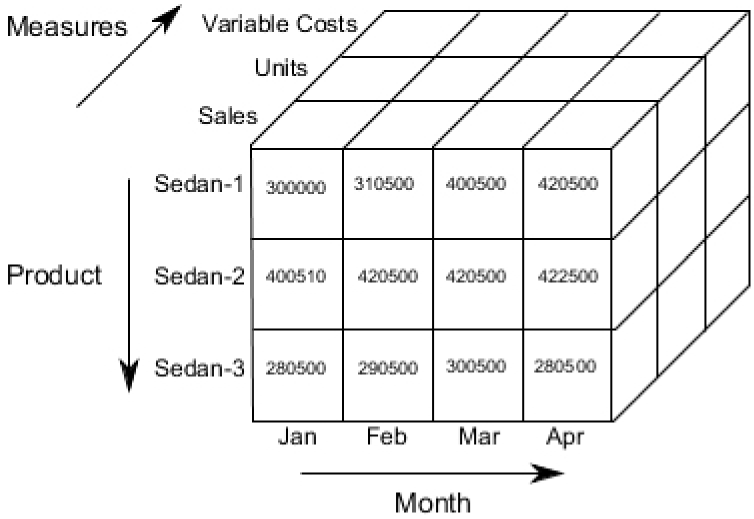
`['size',{'sweaters','vests','jackets'}]=`
`S:('static:XS:S:M:L:XL');`
	This rule statement indicates that any cell identified by the member **size** and any of the members **sweaters**, **vests**, or **jackets** displays a static pick list with the values XS, S, M, L, and XL.

`['fabric']=S:('dimension:materials');`	This rule statement indicates that any cell that is identified by the member **fabric** displays a pick list with all members in the materials dimension.

Cubes

IBM Planning Analytics stores the data that you need for planning and analysis in cubes.

Each cube typically has a specific purpose. Suppose that you are analyzing sales; you have a cube that measures the sales for Sedan cars over time. The cube contains three dimensions: Measures, Product, and Month. Each measure, such as Sales, is organized by a product and a month. For example, the cell value 300000 represents the sales of Sedan-1 in the month of January (Jan).



A cube has two or more dimensions. The number of dimensions that there are in a cube depends on the purpose of a cube. For example, a two-dimensional cube is useful as a lookup table; you can store exchange rates in a lookup table.

A revenue planning cube might have the following dimensions: Products, Versions, Regions, Measures, Time.

Guidelines for designing cubes

1. List the measures you want to track in your business analysis. Examples of measures include sales amounts, units sold, expenses, acquisition values, and campaign costs.
2. Decide what dimensions you need. Consider the following questions:
   - In most analyses, you track measures over time. What is the base time interval: days, weeks, months?
   - Is there a geographic dimension?
   - Do the measures vary by customer and product?
   - Is there a scenario, or versions dimension. For example, do you want to see the actual figures versus the budget?
3. Determine the hierarchical structure of your dimensions. Every dimension has at least one hierarchy. If multiple hierarchies are enabled, you can use them to see alternative rollups of the data in the same view.
4. Create a list of attributes you want to associate with the dimension members. Examples of attributes include store square footage, customer IDs, and local language versions of member names.
5. Define the display formats for the measures in your cubes. For example, define Gross Margin as a percentage and Sales as a currency amount.
6. The dimension order in a cube is important. The order that you select the dimensions when you create a cube can impact performance. In general, select the dimensions in order of the smallest, sparse dimension first, and the largest, dense dimension last. A dense dimension has a high percentage of values for its members. However, you might find that it is better to put a small dense dimension, such as a versions dimension, before a large, sparse dimension such as a products dimension.
7. If the dimension contains members that have the text type, or is a pick list, this dimension must be last in the cube. Text type members and pick list members must be in the same dimension.

For more information about cubes and cube design considerations, see Designing Cubes in the TM1 for Developers documentation.

Create a cube

You can create cubes when you are a Modeler or an Administrator. Cubes are created and stored on a specific database.

Procedure

1. Make sure that you are in Edit mode.
2. In the Data tree, go to the database where you want to create the cube.
3. Expand the database to show the dimensions, cubes, and other associated items.
4. Right-click the **Cubes** group, then click **Create cube**.
5. You can change the database where the cube is stored by selecting a database from the list in the **Create cube** dialog box. The current database is selected by default.
6. On the **Create cube** dialog box, type a name for the cube.

   The cube name can contain spaces, but cannot contain any of the following characters: `\ / : * ? " < > | ' ; ,`
7. In the **Available dimensions** list, select the dimensions to be included in the cube.

   Use **CTRL+click** (Windows) or **CMD+click** (Mac) to select multiple non-consecutive dimensions. Use **SHIFT+click** to select a consecutive group of dimensions.
8. Click → to move your selections to the **Cube dimensions** list.
9. Repeat steps 7 and 8 until all the dimensions you want are in the **Cube dimensions** list.

   If you want to remove any dimensions from the **Cube dimensions** list, select the dimension, then click ←.
10. To change the position of a dimension within the cube, click the dimension in the **Cube dimensions** list, then click the ↑ or ↓ .

    The order of dimensions in a cube can impact system performance. To learn more, see Ordering Dimensions in a Cube in the *TM1 Developer* documentation.
11. Click **Create** to create the cube.

Results

A cube view is placed on the sheet so that you can check that the cube is correct.

Create a cube from a file

Modelers can create and populate a cube and its dimensions from a text file.

Note: If you create a new cube from a text file, you cannot include existing dimensions. All dimensions must be new.

Before you begin

You must be a member of the ADMIN group for the TM1 database to be able to import data into cubes. See “Add users to user groups” on page 241.

Your import data must be correctly structured and formatted to get the right results, see “Cube import file format” on page 221.

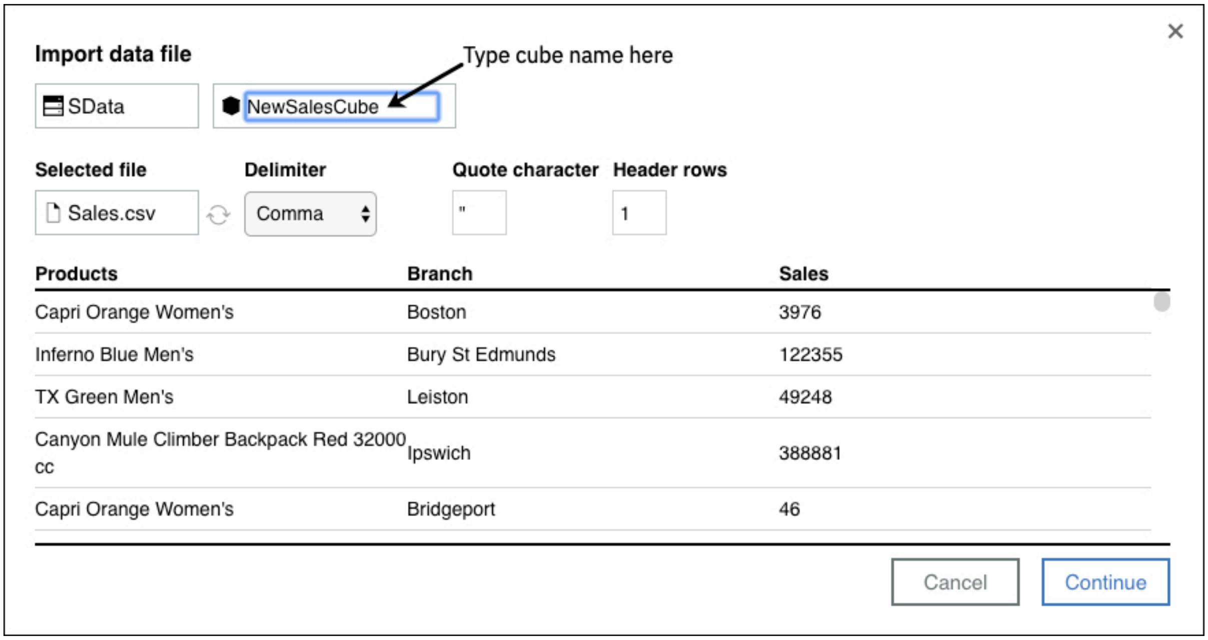
Procedure

1. In edit mode, go to the database in the Data tree where you want to create the cube.
2. Right-click the database, then click **Import data**.
3. Click the cubes list, and select **New cube**.

4. Drag the source file onto the import area, or click the import area to browse for a file, then click **Continue** to import the file.

   A preview of the dimensions in the cube appears.

5. Name the cube in the **New cube name** field.



Import data file

SData | NewSalesCube

Selected file	Delimiter	Quote character	Header rows
Sales.csv	Comma	"	1

Products	Branch	Sales
Capri Orange Women's	Boston	3976
Inferno Blue Men's	Bury St Edmunds	122355
TX Green Men's	Leiston	49248
Canyon Mule Climber Backpack Red 32000 cc	Ipswich	388881
Capri Orange Women's	Bridgeport	46

6. Check that the **Delimiter, Quote character**, and **Header rows** values match what is in your import data file.

   If you change these values, click **Refresh** ⟳.

7. Click **Continue**

8. Name your dimensions in the **Cube dimensions** column.

   The order of dimensions in a cube can impact system performance. To learn more, see Ordering Dimensions in a Cube in the *TM1 Developer* documentation.

   The dimension name must be unique in the database.

   For example, an import file has three columns as shown in the following table.

Products	**Branch**	**Sales**
Capri Orange Women's	Boston	3976
Inferno Blue Men's	Bury St Edmunds	122355
TX Green Men's	Leiston	49248

   You might name your first dimension Products and your second dimension Branch. The third column contains data.

   Note: To add extra dimensions, click **Add dimension**.

9. Select the dimension structural type for each dimension.

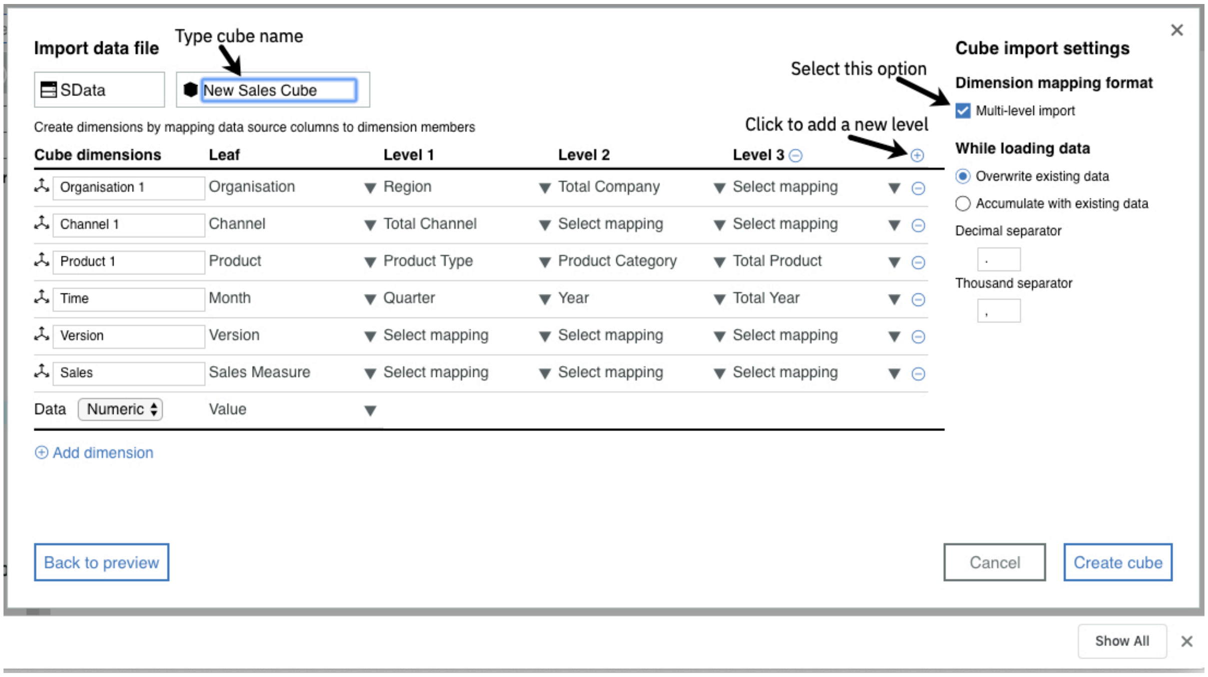
   - For a flat list structure, select **Leaf only** in the **Type** column. For example, you might have a list of products.
   - For a simple parent-child structure, select **Parent-Child** in the **Type** column. For example, you might have two columns, Regions and Cities, where Regions is the parent, and Cities is the child.
   - If you are importing any dimensions with more than two levels, select the **Multi-level import** checkbox under **Cube import settings**. An example of a multi-level dimension is one with the following columns: Product, Product type, Product category, Total product.

10. Map the data source column to dimension members for each dimension.

    - For **Leaf only** dimensions, in the **Child** column, select the column of data, which contains the leaf members.
    - For **Parent-Child** dimensions, in the **Parent** column, select the column of data that contains the parent members. Then, in the **Child** column, select the column of data that contains the child members.
    - For **Multi-level** dimension structures, do the following steps

      a. In the **Leaf** column, select the lowest level item in the dimension structure, for example **Product**.

      b. In the **Level** columns, select the next levels, for example, **Product type** in level 1, Product Category in Level 2. To add more levels, click (+).



Type cube name

Select this option

Click to add a new level

Import data file

SData

New Sales Cube

Create dimensions by mapping data source columns to dimension members

Cube dimensions	Leaf	Level 1	Level 2	Level 3
Organisation 1	Organisation	Region	Total Company	Select mapping
Channel 1	Channel	Total Channel	Select mapping	Select mapping
Product 1	Product	Product Type	Product Category	Total Product
Time	Month	Quarter	Year	Total Year
Version	Version	Select mapping	Select mapping	Select mapping
Sales	Sales Measure	Select mapping	Select mapping	Select mapping
Data Numeric	Value			

Cube import settings

Dimension mapping format

☑ Multi-level import

While loading data

● Overwrite existing data

○ Accumulate with existing data

Decimal separator

.

Thousand separator

,

11. To specify the data type, select the following options in the **Data** row:

    a) Select **Numeric** or **String**.

    b) Select which column the data is mapped to by clicking **Select mapping**.

12. Set the following options as needed:

    Overwrite existing data

    If there is more than one value in the import file for a cell, only the last entry in the import file is imported. This option has no impact on string cells. If the cell contains text, the current value is always overwritten.

    Accumulate with existing data

    Sum up values as they are imported. **Accumulate** has no impact on string cells. If the cell contains text, the current value is always overwritten.

    Decimal separator

    You can set the character to use as the decimal separator.

    Thousand separator

    You can set the character to use as the thousand separator.

13. Click **Create cube**.

What to do next

To check your cube, create a cube view from the cube in the Data tree.

Import data into a cube

Modelers can import data into a cube. Drag a text file onto the cube in the content tree, or right-click the cube in the content tree and select **Import data**.

You can create a process and a chore so that you can schedule regular updates to the data in the cube, and you can import data into control cubes.

You can also create a new cube and dimensions from a text file. See "Create a cube from a file" on page 216.

Note: In IBM Planning Analytics Workspace Local, drag and drop requires Planning Analytics Administration agent to be configured and running.

Before you begin

You must be a member of the ADMIN group for the TM1 database to be able to import data into cubes. See "Add users to user groups" on page 241.

Your import data must be correctly structured and formatted to get the right results, see "Cube import file format" on page 221.

Procedure

1. In edit mode, drag the text file onto the cube in the Data tree.

   You can also right-click the cube in the Data tree and select **Import data**. An **Import data** window opens. You can drag the file onto the import area, or click the import area to browse for a file then click **Continue** to import.
2. Check that the **Delimiter, Quote character,** and **Header Rows** values match what is in your import data file.

   If you change these values, click **Refresh** ⟳.
3. Set the following options as needed.

   Clear data

   Select **Clear data** to reset values in the cube to 0.

   The default option is **None**. Select to prevent data in the cube from being cleared. You can also specify **Whole cube** or **View**. If you select **View**, you must choose which view that you want to clear.

   View to clear

   Select which view you want to clear.

   Accumulate

   Select **Accumulate** to sum up values as they are imported.

   Accumulate adds the values that are imported to the existing contents of the cell.

   If there is more than one value in the import file for a cell, all the values are imported and added to the existing value in the cell.

   Accumulate has no impact on string cells. If the cell contains text, then the current value is always overwritten.

   Overwrite

   Select this option to overwrite existing values in the cube.

   If there is more than one value in the import file for a cell, only the last entry in the import file is imported.

   Overwrite has no impact on string cells. If the cell contains text, the current value is always overwritten.

   Do not clear data

   Clear all data from cube

   Resets all data in the cube to 0.

Clear data from view
Resets all data in the view to 0. Select which view to clear data from.

Overwrite existing data
Overwrites existing values in the cube. If there is more than one value in the import file for a cell, only the last entry in the import file is imported. This option has no impact on string cells. If the cell contains text, the current value is always overwritten.

Accumulate with existing data
Sum up values as they are imported, adding the values that are being imported to the existing contents of the cell. **Accumulate** has no impact on string cells. If the cell contains text, the current value is always overwritten.

Decimal separator
You can set the decimal separator.

Thousand separator
You can set the thousand separator.

4. Map the data source columns to the cube dimension. Click the arrow ▼ in the next to **Select mapping** and select the column for each dimension.
5. To create a process that contains the options that you specified, select the **Save as process** option, and name the process.

   The process is saved in the **Processes** branch of the content tree. You can modify the process, see "Create and edit processes" on page 248.
6. To schedule the process that you created, select the **Schedule process as** option, and name the chore.

   A chore is created and opened on the tab so that you can set the schedule, see "Create and edit chores" on page 259. The chore is saved in the **Chores** branch of the content tree.
7. Click **Import**.

Results

A message displays how many lines of data are successfully imported. If some data cannot be imported, the details are displayed. Some common errors are shown:

```
line (300): Invalid key: Dimension Name: "Products", Element Name (Key): "Mountain Man Analog"
line (293): Cell type is consolidated
line (5000): Cannot convert string "186432x" to a real number
```

Invalid key
Invalid key means that the data in the import file can't be matched up with the data in the cube.

In the example:

```
line (300): Invalid key: Dimension Name: "Products", Element Name (Key): "Mountain Man
Analog"
```

The member name "`Mountain Man Analog`" doesn't exist in the Products dimension.

To fix this error, you can either add the member to the Products dimension, or you can update the import file. Or if this line isn't important, you can ignore the error.

`Cell type is consolidated`
You can't import data into a consolidated cell.

Cannot convert string "186432x" to a real number
A numeric value is expected, but the value contains a non-numeric character.

Cube import file format

When you import a file into a cube, the file must be structured and formatted in a certain way to get the results that you want.

File formats

The import data must be saved in a text file format, such as comma-separated (CSV) or with any other standard delimiter such as tab, space, semicolon, colon, vertical line, or caret (^).

Note: Files that contain a Byte Order Marker (BOM) might be corrupted when the file is imported.

Structure of the data

Each column in the text file represents either a dimension or the values.

Value data

Value data is the data that goes into cells. Values can either be numbers or string values. You can import only one column of value data.

Dimensions

The items in the dimension columns must match up with the dimension member names in the cube, otherwise the row is not imported, and an error such as

```
line (300): Invalid key: Dimension Name: "Products", Element Name (Key): "Mountain Man
Analog"
```

is displayed.

If you are creating a new cube, the file must contain one dimension per column.

The column headings do not need to match the dimension names.

The column order in the text file does not need to match the order of dimensions in the cube, and you select which columns of data to import. The column headings do not need to match the dimension names.

If several rows in the text file target the same cell, the value in the last row is imported, and all other values are ignored unless the **Accumulate with existing data** option is selected. **Accumulate with existing data** sums up values as they are imported, adding the values that are being imported to the existing contents of the cell.

Importing data into a cube

Say that you have a cube with three dimensions: Cities, Products, Time_months. You want to import a .csv text file with five columns that contains sales data. The first eight lines of the text file, including the header row, are shown.

```
Product,Month,Branch,Qty,Sales
Capri Orange Women's,Nov-15,Boston,3976,3976
Inferno Blue Men's,Jul-15,Bury St Edmunds,2719,122355
TX Green Men's,Aug-15,Leiston,3078,49248
Canyon Mule Climber Backpack Red 32000 cc,Apr-15,Ipswich,4801,388881
Capri Orange Women's,Sep-15,Bridgeport,46,46
Glacier Deluxe Black Unspecified,Sep-15,London,3839,122848
Cat Eye Orange Women's,Jun-15,Dundee,3221,235133
```

1. The header row contains five columns: Product, Month, Branch, Qty, and Sales
2. Map the following columns to the dimension names as listed:
   - Map the Product column to the Products dimension
   - Map the Month column to the Time_months dimension
   - Map the Branch column to the Cities dimension
3. Decide which values you want to bring in, either Qty, or Sales. In this case you want to import the Sales data, so select **Numeric Values** for Sales. The Qty column of data is ignored.
4. You can now import the file.

Creating a cube with multi-level dimensions

The following example CSV file has 16 columns of data. Some of the dimensions have 3 or more levels.

```
Organization, Region, Total Company, Channel, Total Channel, Product, Product Type, Product
Category, Total Product, Year, Month, Quarter, Total Year, Version, Sales Measure, Value
Massachusetts, East Region, Total Company, Retail, Total Channel, 3G 32Gb, 3G Smart Phones,
Phones, Total Product, 2018, Jan, Q1, Total Year, Budget, Unit Net Sales Price, 70
Florida, East Region, Total Company, Distribution, Total Channel, 3G 16Gb, 3G Smart Phones,
Phones, Total Product, 2018, Sep, Q3, Total Year, Budget, Volume - Units, 5,58
Michigan, Central Region, Total Company, Retail, Total Channel, SP 2110, Desktops, PCs, Total
Product, 2018, Dec, Q4, Total Year, Actual, Direct COGS, 15592
Maryland, East Region, Total Company, Internet, Total Channel, 8 64 Gb, 8 Inch Tablets,
Tablets, Total Product, 2018, Feb, Q1, Total Year, Budget, Volume - Units, 26,97
Illinois, Central Region, Total Company, Distribution, Total Channel, 3G 32Gb, 3G Smart Phones,
Phones, Total Product, 2019, Feb, Q1, Total Year, Actual, Unit Direct Cost, 47,141,464
California, West Region, Total Company, Retail, Total Channel, SP 2110, Desktops, PCs, Total
Product, 2019, Nov, Q4, Total Year, Budget, Indirect COGS, 100
California, West Region, Total Company, Retail, Total Channel, SP 2110, Desktops, PCs, Total
Product, 2019, Nov, Q4, Total Year, Actual, Unit Net Sales Price, 341454
```

To create a new cube and dimensions with this file, do the following steps:

1. Select the **Multi-level import** option.
2. Click + to add a level.
3. Click **Add dimension** four times.
4. Map the data as shown in the following table. You might need to change the names in the cube dimensions name if dimensions exist with that name.

Cube dimensions	Leaf	Level 1	Level 2	Level 3
Organization	Organization	Region	Total Company	
Channel	Channel	Total Channel		
Product	Product	Product Type	Product Category	Total Product
Time	Month	Quarter	Year	Total Year
Version	Version			
Sales	Sales Measure			
Data	Numeric	Value		

5. Click **Create cube**.

Change the order of dimensions in a cube

You can change the order of dimensions in a cube in IBM Planning Analytics Workspace, and view the percent change in memory. Changing the order can improve the performance of a model, by optimizing dimension order to consume less memory.

When you optimize the order of dimensions in a cube, Planning Analytics does not change the actual order of dimensions in the cube structure. Instead, it changes the way dimensions are ordered internally on the database. Because the cube structure is not changed, any rules, functions, or applications that reference the cube remain valid.

As you change the order of dimensions, you can instantly view a report detailing the impact that your changes have on cube memory consumption.

For the following reasons, experiment with the order of dimensions in a cube only in a development environment:

- Significant memory resources are required for the database to reconfigure the order of dimensions in a cube. During the reordering process, the temporary RAM on the database increases by a factor of two for the cube that you are reordering. For example, a 50 MB cube requires 100 MB of RAM to reconfigure.
- Reordering puts a read lock on the database, locking all user requests while the reorder is performed.

When you have a satisfactory dimension order in your development environment, modify the order of dimensions in your production environment. Remember that there will be an impact on memory consumption and server availability while the reorder is being executed.

About this task

You can change the order of dimensions for cubes on Planning Analytics databases version 2.0.6 or later. If a cube is on a database before version 2.0.6, the **Reorder dimensions** option is unavailable when you right-click the cube in the content tree.

You must be a Planning Analytics administrator or modeler to optimize the order of dimensions in cubes. When you optimize the order of dimensions in a cube, you cannot move a string dimension from the last position. If a string dimension exists in your cube, it must be the last dimension in the ordering.

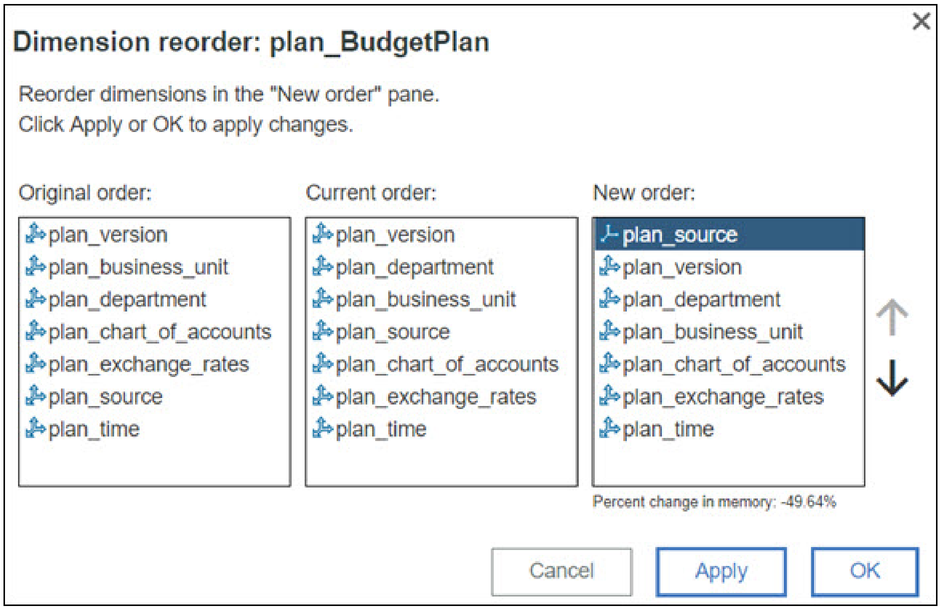
You cannot reorder dimensions in control cubes.

Procedure

1. In the Data tree, right-click the cube that you want to optimize.
2. Click **Reorder dimensions**.

   The **Dimension reorder** screen opens. This screen shows the original order of dimensions, the current ordering of dimensions on the database, and a list of cube dimensions that you can manipulate to set a new order.
3. Select a dimension in the **New order** list.
4. Click the up ↑ or down ↓ arrow to change the order of the dimension in the cube.
5. Click **Apply**. If prompted for confirmation, click **OK**.

   Note the value next to the **Percent change in memory** label. If this value is negative, the new order of dimensions uses less memory and is therefore more efficient.



Dimension reorder: plan_BudgetPlan

Reorder dimensions in the "New order" pane.
Click Apply or OK to apply changes.

Original order:	Current order:	New order:
plan_version	plan_version	plan_source
plan_business_unit	plan_department	plan_version
plan_department	plan_business_unit	plan_department
plan_chart_of_accounts	plan_source	plan_business_unit
plan_exchange_rates	plan_chart_of_accounts	plan_chart_of_accounts
plan_source	plan_exchange_rates	plan_exchange_rates
plan_time	plan_time	plan_time

Percent change in memory: -49.64%

6. Repeat steps 3 - 5 until you achieve the most efficient ordering of dimensions.
7. Click **OK**.

Lock and unlock a cube

Lock a cube to prevent updates to the data in the cube.

Analysts with lock permissions can lock a cube. You must be a member of the default Admin or DataAdmin groups to unlock a cube.

Procedure

1. Right-click the cube in the Data tree and select **Lock cube**.

   A locked cube has the lock icon. The cells in a locked cube view are gray when the view is refreshed.
2. To unlock the cube, right-click the cube and select **Unlock cube**.

Unload and reload a cube from memory

You can unload a cube from memory to temporarily reduce RAM consumption or to assist in the development and troubleshooting of rules feeders. Unloading a cube also unloads any views of the cube from memory.

About this task

While a cube is unloaded, any reference to or request for data in the cube will cause the cube to be automatically reloaded, maintaining data availability.

You must be an administrator or modeler to unload or reload cubes.

Procedure

1. Make sure that you are in edit mode.
2. In the Data tree, right-click the cube you want to unload, then click **Unload cube**.
3. Click **OK** when prompted for confirmation.

   To manually reload a cube into memory, right-click the cube, then click **Reload cube**.

Delete a cube or a view

You can delete cubes or views from the content tree when you are a Modeler or an Administrator.

About this task

You cannot delete control cubes, which are used to run and maintain your IBM Planning Analytics system. When you delete a cube, you also delete all data and any rules or views that are associated with the cube.

Procedure

1. In the Data tree, navigate to the database that contains the cube or view that you want to delete.
2. Expand the database to reveal the dimensions, cubes, views, and other groups.
3. To delete a cube, click **Cubes**, right-click the cube that you want to delete, then click **Delete cube**.
4. To delete a view from the database, click **Views**, right-click the view that you want to delete, then click **Delete view**.

Manage cube settings

Modelers can view and edit cube security, properties, and attributes in the settings editor.

Procedure

1. Expand the database that you want to manage the settings for in the tree, and right-click **Cubes**.
2. Click **Edit settings**.
3. Click 🏷to view cube attributes, click 🔒 to view cube security settings, and click ☰ to view cube properties.

What to do next

To find out more about cube security settings, see "Secure cubes, dimensions, processes, and chores" on page 243.

Cube properties

Cube properties provide information about cubes that are used in different processes. Some of these properties cannot be edited in the settings editor. They are updated by the system.

Element	Description
REPSTATUS	If a cube is replicated on the database, the value of RepStatus is `Copied`, otherwise this property value is empty. For more information, see Replicating Cubes(https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_op.2.0.0.doc/c_replicatingcubes_n110007.html)..
SYNCVIEWS	For replicated cubes, this property indicates whether views are synchronized when the associated cube is synchronized.
SYNCRULE	For replicated cubes, this property indicates whether rules are synchronized when the associated cube is synchronized.
DEMANDLOAD	Indicates that a cube is automatically loaded when the database starts or is loaded 'on demand' only when a cube value is requested. By default, all cubes are loaded into memory when a database starts. While this allows fast access to data, it can consume significant resources. If your database contains infrequently accessed cubes, you can conserve resources by setting those cubes to load only when a client attempts to access the cube data. When a cube is loaded on demand, the value of the DemandLoad property is YES, otherwise the property value is NO.
TIME_DIMENSION	If a time dimension is defined for a cube, this property stores the name of the dimension.
MEASURES_DIMENSION	If a measures dimension has been defined for a given cube, this property stores the name of the dimension.
LOCK	When a cube is locked, this property stores the name of the client that has locked the cube.
VMT	For each cube, this property defines the time threshold, in seconds, beyond which the algorithm that stores Stargate views is triggered. If the time required to calculate a cube view surpasses the specified threshold, IBM Planning Analytics attempts to store a Stargate view. If not enough memory is available to store the Stargate view, Planning Analytics purges the oldest Stargate view that is not currently in use, and continues to purge views in this manner until sufficient memory is made available. If no VMM value is specified the default value is 5 seconds. The valid range is 1 - 259,200 seconds.

Element	Description
VMM	For each cube, this property determines the amount of RAM reserved on the server for the storage of Stargate views. The more memory made available for Stargate views, the better the performance is. You must, however, make sure sufficient memory is available for the TM1 server to load all cubes. The value of VMM is expressed in kilobytes. If no VMM value is specified the default value is 128 kilobytes. The valid range for VMM is 0 - 2,147,483,647 KB. The actual upper limit of VMM is determined by the amount of RAM available on your system.
REPSCRNAME	For replicated cubes, this property stores the name of the source cube.
REPLICATION	For replicated cubes, this property stores the name of the replication connection with which the replicated cube is associated.
LOGGING	Indicates that logging is enabled for a cube. When cube logging is enabled, the value of this property is YES, otherwise the value is NO.
VIRTUALCUBE	Stores the name of the virtual cube. A virtual cube has no data loaded in it, but references data in other cubes by using rules.
PROVIDER	Stores the name of the ODBO provider that is used when data is imported from an OLE DB for OLAP (ODBO) data source. For example, TM1 uses "TM1 OLE DB MD Provider" and Microsoft Analysis Services uses "Microsoft OLE DB Provider for OLAP Services 11.0"
LOCATION	Stores the name of the location for a particular instance of the ODBO provider service. Used when data is from an OLE DB for OLAP (ODBO) data source.
DATASOURCE	Stores the name to a set of catalogs at a particular location. In Microsoft Analysis Services, this is the name of a registered server. Used when importing data from an ODBO data source.
CATALOG	Stores the name to a collection of databases (cubes, dimensions, and other objects). For Microsoft Analysis Services, this is the name of the database. Used when importing data from an ODBO data source.
USERID	Stores the TM1 user name, for example, Admin. Used when importing data from an ODBO data source.
PASSWORD	The password for the TM1 user. Used when importing data from an ODBO data source.
SAPCLIENTID	Used when connecting to SAP. The client number that corresponds to the UI version on the SAP server to which you are connected.
SAPCLIENTLANG	Used when connecting to SAP. The language setting for the SAP system.
PROVIDERSTRING	
SAPVARIABLESCLAUSE	
SLICERMEMBERS	

Element	Description
DATARESERVATIONMODE	Data Reservation (DR) is a server-related feature in TM1 that allows you to configure exclusive write-access to regions of a cube for individual users. By default, Data Reservation is not enabled. An administrator must enable and configure the feature before you can use the related TurboIntegrator and API functions to manage Data Reservations. You can enter the following values in English only. These keyword values are not translated: **OFF** Turns off the Data Reservation feature for the specific cube. The default value is OFF. **REQUIRED** This mode disables write access for all users for the entire cube and requires you to explicitly assign Data Reservations for any user that needs to write to this cube. For example, a user must have a Data Reservation on a cell if they want to write to that cell. **REQUIREDSHARED** This mode is a variation of the REQUIRED mode that allows Data Reservations for different users to overlap. All other aspects of this mode behave the same as REQUIRED mode. REQUIREDSHARED mode was implemented to accommodate overlapping requests that use multi-node edit capability in IBM Cognos TM1 Applications. This mode is the default assigned DR mode on all cubes represented by Cube Views or Manual Dependencies in TM1 Applications. In REQUIRED mode, the TM1 database restricts write access to a slice by allowing only a single user to have a reservation for a node at any one time. In REQUIREDSHARED mode the application must enforce this restriction if necessary. **ALLOWED** Maintains write access, based on security, for all users across the entire cube, but allows you to selectively restrict write access to an area of the cube by assigning Data Reservations to individual users. For example, ALLOWED mode sets aside a section of a cube for a specific user while keeping write access available for all other users to the rest of the cube.
ALLOWPERSISTENTHOLDS	Specifies whether cells continue to be excluded from data spreading after the user logs out and logs in again.
CALCULATIONTHRESHOLD	Sets the threshold for the calculation cache in terms of number of calculation steps. Setting the cube property overrides the global value CalculationThresholdForStorage that is defined in the `tm1s.cfg` file.
RULE_STATS	This property determines whether performance statistics are collected for the rules that are associated with the cube. To enable statistic collection, set the Rule_Stats property to YES. To disable statistic collection, set the property to NO, the default value). Rule_Stats is a dynamic property, meaning that it does not require a server restart to take effect. However, it can take up to 60 seconds for a dynamic property change to be applied on the TM1 server.

Element	Description
MAINTAIN_DETAILED_ FEEDER_MEMORY_STATS	A value of 'YES' for the (cubeName, propertyName) tuple turns on the detailed feeder memory used calculation. **Note:** The detailed feeder memory used calculation is intended to be used during pre-production as an aid to help design the cube. When this step is complete, the property value should be switched to 'NO', so that the much faster feeder memory used calculation is used during production.

Cube attributes
Cube attributes provide information about the purpose of cubes.

An example of a cube attribute is the CUBE_TYPE attribute. This attribute is used to indicate that a cube is a metrics cube. The value is METRICS. Another example of a cube attribute is CAPTION. The CAPTION attribute is used to enable cube names to be displayed in different languages.

You can view and add values to cube attributes in the settings editor, but you cannot add new cube attributes in the setting editor. To add new cube attributes, create a process.

If no cube attributes defined, or you don't have the correct permissions to view the cube attributes, a message is displayed. If you don't have the correct permissions, contact your administrator.

The following example creates Caption as an alias (A) attribute and adds French (fr) as a language, and the French translation `Ventes` for the Sales cube.

```
CubeAttrInsert( '', 'Caption', 'A');
CubeAttrPutS( 'Ventes', 'Sales', 'Caption', 'fr' );
```

To find out how to create a process, see “TurboIntegrator processes” on page 248. To find out more about the Caption attribute, see “Captions for members” on page 210.

Rules

You can create complex data calculations in IBM Planning Analytics with business rules.

The most common calculations in Planning Analytics involve aggregating data along a dimension. These calculations are performed automatically based on your dimension hierarchy definitions. However, in many models you need to perform calculations that do not involve aggregating, such as cost allocations and exchange translations. With business rules, you can create formulas to perform these calculations.

View this video to learn the basics about business rules in Planning Analytics Workspace.

https://youtu.be/9YG6wkJRmvg

With business rules, you can perform the following tasks:

- Multiply prices by units to yield the sales amounts.
- Override consolidations when necessary. For example, you can prevent a quarterly price from displaying a tally of individual monthly prices.
- Use data in one cube to perform calculations in another cube, or share data between cubes. For example, you can pull sales data into a cube that contains Profit and Loss information.
- Assign the same values to multiple cells.

For more information about business rules, see the *TM1 Rules* documentation.

Create and edit rules

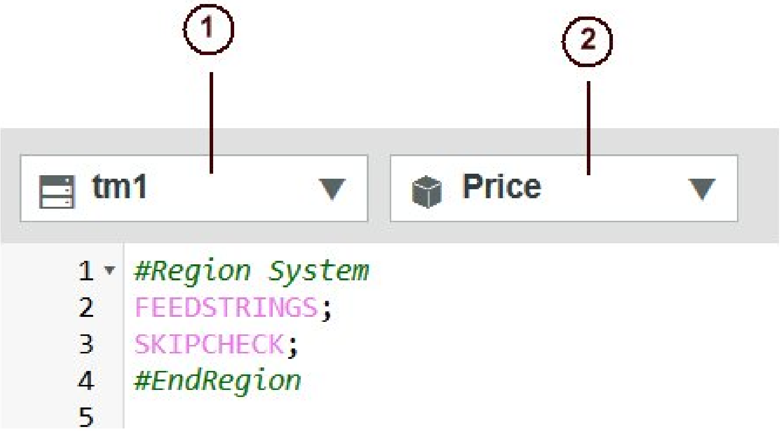
You can create and edit business rules when you are a Modeler or an Administrator.

Procedure

1. Open Planning Analytics Workspace and log in with a user that has the Modeler or Administrator role.

2. Make sure that you are in edit mode to create or edit a rule.
3. In the Data tree, expand to the database where you want to create a business rule, and the **Cubes** group.
4. Right-click the cube for which you want to create or edit a rule and click **Edit business rules**.
5. Create or edit your rules in the Rules Editor.

   When the Rules Editor is open, you can use the database and rule lists to quickly open any other rule to which you have access.

   ```
   #Region System
   FEEDSTRINGS;
   SKIPCHECK;
   #EndRegion

   ```

   1

   Database list.

   2

   Rules list.

   To open another rule in the current database, click the rule list and choose the rule to open.

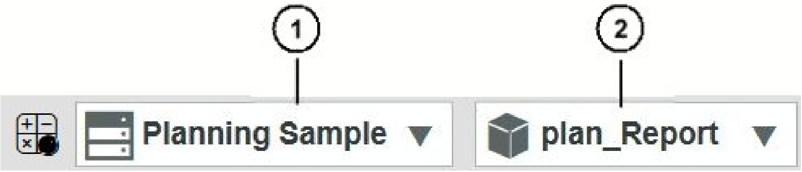
   For more information, see Components of a rule in the TM1 Rules documentation.
6. Click the **Save** icon.

Use the rules editor

You can type business rules directly in the Planning Analytics Workspace rules editor. The editor also provides several features that simplify the process of creating business rules.

Rule switching

You can use the database and rule selectors to open a different rule.



1

Database selector.

2

Rule selector.

To open a new rule on the same database as the current rule, click the rule selector and then click the rule that you want to open.

To open a rule on a different database, click the database selector and pick the appropriate database, then click the rule selector and click the rule that you want to open.

Auto-completion

You can use the CTRL+SPACE keystroke combination to choose from a list of applicable items in the rules editor. If you use CTRL+SPACE after typing one or more characters of an object name, the list presents all items that match the characters you typed.

When you use the auto-completion keystroke combination within an area definition (inside [ ] brackets) or formula component of a rule statement, you are presented with valid selections.

If the space within the brackets is empty, auto-completion presents a list of all dimensions and members that are part of the cube that is associated with the rule. Dimensions are listed first in alphabetical order, followed by all members from all dimensions in alphabetical order. Members are not grouped according to the dimensions to which they belong.

A fully qualified member reference in an area definition or formula component contains three parts: a dimension specification, a hierarchy specification, and the member name.

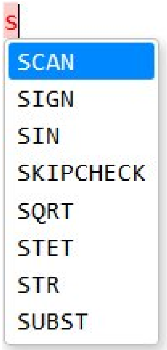
```
DimName:'HierarchyName':'MemberName'
```

If hierarchy support is not enabled in your database, the hierarchy specification is not necessary. All three parts are not necessary to create a member reference; it is usually sufficient to use the member name. However, if you are creating a fully qualified reference, auto-completion presents you with just the items that are relevant to the current partial reference.

For example, if you enter a dimension name in the area definition, then use the CTRL+SPACE keystroke combination, you see a list of the hierarchies and members that are part of the dimension.

Similarly, if your area definition includes a dimension name and a hierarchy name, you see a list of the members that are part of the hierarchy when you use the auto-completion feature.

If you begin entering a function name in the rules editor outside of an area definition or formula component, and then use the CTRL+SPACE keystroke combination, you can pick from a list of rules functions that match the characters you typed.



Drag from the Data tree

You can drag dimensions, hierarchies, and members from the Planning Analytics Workspace data tree into the rules editor.

When you drag an object from the data tree and drop it on a portion of the rules editor that is not within an area definition, a new area definition is created with a reference to the object:

- For a dimension, the result is `[DimName:]` with the DimName quoted when required by TM1.
- For a hierarchy, the result is `[DimName:'HierarchyName':]`
- For a leaf member, the result is `['MemberName']`, unless the member name is not unique across all dimensions in the cube. If the leaf member name is not unique, the result is a fully qualified reference such as `[DimName:'HierarchyName':'MemberName']`. For all consolidated members, regardless of uniqueness across dimensions, the result is a fully qualified reference.

When you drag an object from the data tree and drop it on a portion of the rules editor that is within an area definition, the area definition is updated to include the object.

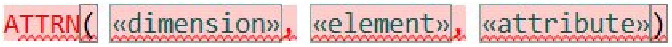
- If the area definition does not include a reference to a dropped dimension, then `,DimName:` is appended to the end of the area definition. If the area definition already includes a reference to a dropped dimension, then the reference is not updated.
- If the area definition does not include a reference to a dropped hierarchy, then `,DimName:'HierarchyName':` is appended to the end of the area definition. If you attempt to drag a hierarchy that is not valid for the current area definition, you receive an error.
- If the area definition does not include any reference to a member, then dropping a member onto the area definition adds the member. If the area definition already includes a reference to a single member in a dimension, then dropping another member from the same dimension updates the reference to include all members in subset notation. For example, `[DimName:'HierarchyName':{'MemberName1', 'MemberName2'}]`. If you drop any additional members from the same dimension, the member names are added to the subset notation.

Function list

You can click *fx* on the rules editor to quickly insert a rules function into your business rules.

When you click *fx*, you see a list of function categories. Click a function category to see the available functions within the category, then double-click a function to insert it at the cursor in the rules editor.

The function is inserted with placeholders for required parameters. You must replace the placeholders with valid parameter values. For example, if you use the function list to insert the ATTRN function into your rules, you see this message:



```
ATTRN( «dimension», «element», «attribute»)
```

You must replace the `<<dimension>>`, `<<element>>`, and `<<attribute>>` placeholders to successfully return a numeric attribute value with ATTRN.

Shortcut keys

There are many shortcut keys available in the rules editor. You can use these keys to edit statements, find and replace text, and navigate through the statements in the rules editor.

To see a full listing of shortcut key combinations, click ⋮, then click **Display shortcut keys**.

To hide the listing of shortcut keys, click ⋮, then click **Hide shortcut keys**.

Line wrapping

To enable or disable line wrapping in the editor, click ⋮, then click **Enable line wrapping** or **Disable line wrapping**.

Text customization

To change the font and size of the text that is displayed in the rules editor, click ⋮, then click **Change Font**.

Delete rules

You can delete business rules when you are a Modeler or an Administrator.

Delete rules in Planning Analytics Workspace

Procedure

1. Open Planning Analytics Workspace and log in with a user that has Modeler or Administrator permissions.
2. Make sure that you are in edit mode.
3. In the Data tree, navigate to the database where you want to delete a rule, and expand the database to reveal the dimensions, cubes, and other associated object groups.
4. Expand the **Cubes** group, and the cube for which you want to delete rules.
5. Right-click the rule and select **Delete business rules**.

Delete rules in Planning Analytics Workspace Classic

Procedure

1. Open Planning Analytics Workspace and log in with a user that has Modeler or Administrator permissions.
2. Make sure that you are in edit mode.
3. In the Data tree, navigate to the database where you want to delete a rule, and expand the database to reveal the dimensions, cubes, and other associated object groups.
4. Expand the **Cubes** group, and the cube for which you want to delete rules.
5. Right-click the rule +−×÷and select **Delete business rules**.

Trace cell values

If a cell value is derived through consolidation or rules, you can review or 'trace' the underlying values that contribute to the consolidation or the rules that define the cell value.

About this task

You can trace cell values only in cubes that reside on a IBM Planning Analytics TM1 database version 2.0.3 or later. If you are viewing a cube on a TM1 database prior to 2.0.3, the **Trace Cell** option is not available on the right-click menu from a cell.

The **Trace Cell** option is available only on cell values that are derived through consolidation or rules. The option is unavailable on leaf cells.

Procedure

1. Right-click the cell you want to trace.
2. Click **Trace cell**.

   The **Trace cell** page displays a table with details about the cell you are tracing. The first row shows information about the original cell from which you initiated tracing. You can click the expand/collapse icon next to the cell coordinates to trace the original cell's value. As you trace the value, more rows appear to provide further details about how the original cell value is derived.

   The table includes the following columns:

   Cell

   For the first row, identifies the original cell by its coordinates in the cube, by using a comma-separated list of member names. As you click the expand icon to trace the original value, more rows show the members that contribute to the original value.

Value

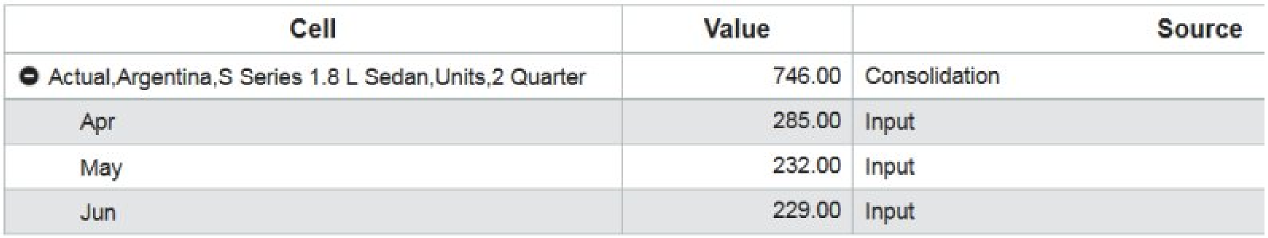
The value of the original cell or member for the current table row. If the value is over 10,000, the value that is displayed is truncated to an integer, otherwise the value shows two decimal places. You can hover over the displayed value to view the actual raw value as it exists in the TM1 database.

Source

Identifies the source of the value: Consolidation, Input (a leaf cell that is not rule calculated), or rule. If the source is a rule, the entire rule is displayed. If the rule is too long to appear in its entirety within the confines of the column, you can click the expand/collapse icon to reveal the full rule.

3. Click the expand/collapse icons in the **Cell** column to trace the original value as far as you want to go.

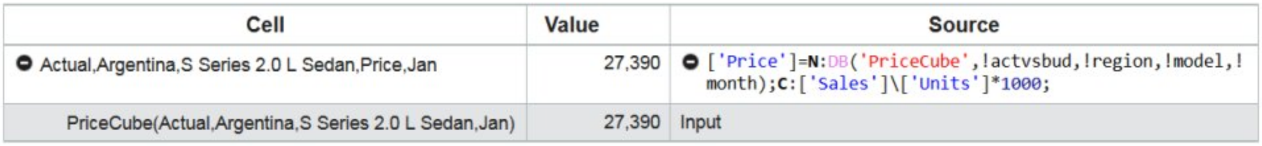
Example



Cell	Value	Source
⊖ Actual,Argentina,S Series 1.8 L Sedan,Units,2 Quarter	746.00	Consolidation
Apr	285.00	Input
May	232.00	Input
Jun	229.00	Input

Figure 1. Simple Consolidation

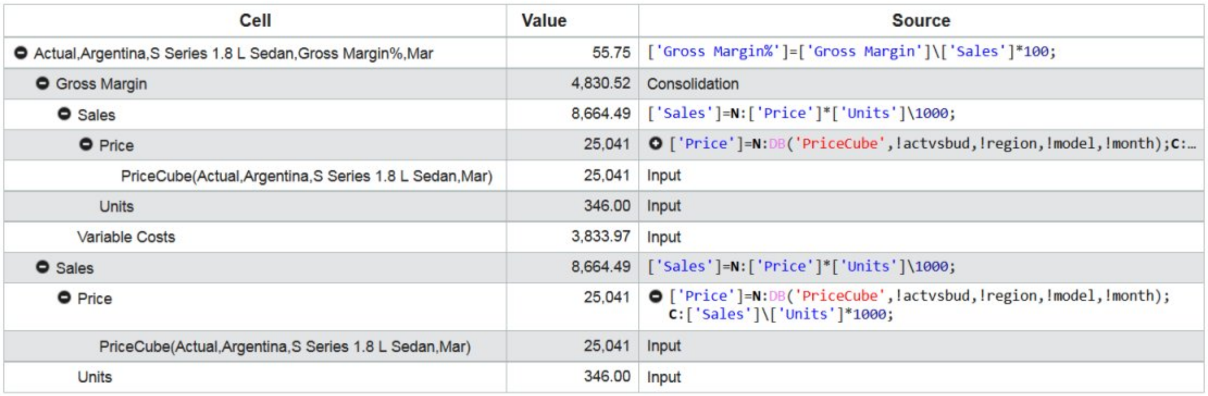
In this example, the cell trace is initiated from a consolidation cell that is identified by the members Actual, Argentina, S Series 1.8 L Sedan, Units, 2 Quarter. When you click the expand icon in the first row of the Cell column, the three input leaf members and their values are revealed.



Cell	Value	Source
⊖ Actual,Argentina,S Series 2.0 L Sedan,Price,Jan	27,390	⊖ ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);C:['Sales']\['Units']*1000;
PriceCube(Actual,Argentina,S Series 2.0 L Sedan,Jan)	27,390	Input

Figure 2. Simple Rule

In this example, the cell trace is initiated from a cell that is derived through rules and identified by the members Actual, Argentina, S Series 2.0 L Sedan, Price, Jan. Because Price is calculated by a rule, the rule appears in the Source column. When you click the expand icon in the first row of the Cell column, the component of the rule that calculates the original cell value is revealed. In this case, the component is a reference to an Input value in the PriceCube.



Cell	Value	Source
⊖ Actual,Argentina,S Series 1.8 L Sedan,Gross Margin%,Mar	55.75	['Gross Margin%']=['Gross Margin']\['Sales']*100;
⊖ Gross Margin	4,830.52	Consolidation
⊖ Sales	8,664.49	['Sales']=N:['Price']*['Units']\1000;
⊖ Price	25,041	⊕ ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);C:…
PriceCube(Actual,Argentina,S Series 1.8 L Sedan,Mar)	25,041	Input
Units	346.00	Input
Variable Costs	3,833.97	Input
⊖ Sales	8,664.49	['Sales']=N:['Price']*['Units']\1000;
⊖ Price	25,041	⊖ ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);C:['Sales']\['Units']*1000;
PriceCube(Actual,Argentina,S Series 1.8 L Sedan,Mar)	25,041	Input
Units	346.00	Input

Figure 3. Combination

In this example, the cell trace is initiated from a cell that is derived through rules. The cell is identified by the members Actual, Argentina, S Series 1.8 L Sedan, Gross Margin%, and Mar. Because Gross Margin% is calculated by a rule, the rule appears in the Source column.

Gross Margin% is calculated by using the members Gross Margin and Sales. When you click the expand icon in the first row of the Cell column, new rows for these members appear in the table. Gross Margin is a consolidation, while Sales is defined by a rule. You can click the expand icon next to both of these members to further trace the values that contribute to the original cell value.

Trace feeders

From an Exploration view, you can right-click a leaf cell and click **Trace feeders** to trace how a selected cell feeds other cells. You can use this capability to debug or improve the performance of your cube rules.

About this task

You can feed only from a leaf member, so the **Trace feeders** option is not available for consolidated cells. However, the option is available for leaf cells defined by rules.

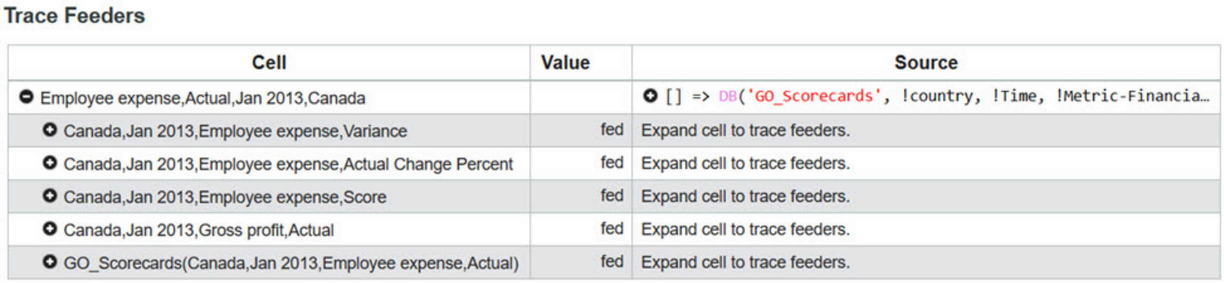
Note: You can trace feeders only from cubes that reside on Planning Analytics database version 2.0.3 or later. If your cube resides on a database before version 2.0.3, the **Trace feeders** option is not available when you right-click a leaf cell.

You can learn more about rules in general, and feeders in particular, in the TM1 Rules Guide.

Procedure

1. In an Exploration view, right-click the leaf cell you want to trace.
2. Click **Trace Feeders**.

   The **Trace Feeders** window opens. When you open the window, the first row in the grid shows the cell from which you initiated feeder tracing. Below the first row you can see the cells that are directly fed by that cell.



Trace Feeders

Cell	Value	Source
⊖ Employee expense,Actual,Jan 2013,Canada		⊕ [] => DB('GO_Scorecards', !country, !Time, !Metric-Financia…
⊕ Canada,Jan 2013,Employee expense,Variance	fed	Expand cell to trace feeders.
⊕ Canada,Jan 2013,Employee expense,Actual Change Percent	fed	Expand cell to trace feeders.
⊕ Canada,Jan 2013,Employee expense,Score	fed	Expand cell to trace feeders.
⊕ Canada,Jan 2013,Gross profit,Actual	fed	Expand cell to trace feeders.
⊕ GO_Scorecards(Canada,Jan 2013,Employee expense,Actual)	fed	Expand cell to trace feeders.

The grid includes three columns: **Cell**, **Value**, and **Source**.

Cell

Identifies the cell by its coordinates in the cube using a comma-separated list of invariant names. If the cell is not from the cube in which feeder tracing was initiated, the cell is identified as `CubeName(Member0, Member1,...)`.

An expand/collapse button appears next to the cell coordinates. You can click the button to reveal the cells that are fed by the current cell, and you can click the button again to hide the cells that are fed by the current cell.

Value

Indicates whether the cell is fed or is unfed.

Source

Displays the feeder statements that are associated with the cell. The statements are formatted and colorized as they appear in the rules editor.

To minimize space, feeders statements are initially shown truncated on a single line. An expand/collapse button appears next to the feeder statements. You can click the button to reveal all the feeder statements that are associated with the cell, and you can click the button again to collapse the statements to a single line.

When you first open the **Trace Feeders** window, the statements for the cells that are fed by the cell from which you initiated feeder tracing are unknown. The Source column displays the message 'Expand cell to trace feeders.' Click the expand/collapse button in the Cell column to calculate the feeder statements. If the cell is included in feeder statements, then the statements are displayed in the Source column and any fed cells appear as child entries. If the cell is not associated with any feeders, the Source column displays the message 'This cell does not feed any cells.' This is also the case as you trace through feeders. Calculation occurs when you click the expand/collapse button in the Cell column.

3. Click the expand/collapse button for any location in the Cell column to reveal how that cell feeds other cells in your model. If the cell feeds other cells, those other cells appear as children in the Cell column. You can continue to trace down through children until you reach the point where a child cell does not feed any other cells.

Configure drill-through to reveal detailed data

You can associate a cube cell with extra data, which can provide underlying detail for the cell or other information relevant to the cell. When you complete the configuration that is required to establish this association, users can drill through to the additional data directly from an Exploration view.

Drill-through configuration consists of two components:

- **Drill rule** - Defines the relationship between the cell and the detailed data.
- **Drill process** - Defines the detailed data that you want to associate with a cell.

When both a drill rule and a drill process are in place for a cell, you can right-click the cell and click **Drill through** to reveal underlying detail. If multiple drill processes are associated with a cell, you can pick the one you want to use.

For more information, see "Drill through to detailed data" on page 126.

Create a drill rule

A drill rule is a unique type of rule that associates view cells with related data. The related data, which is opened through a TurboIntegrator process, can be a cube view or an ODBC source.

About this task

Drill rules are distinct from cube business rules. While business rules define the data for specified cells in a cube, drill rules identify the relationship between specified cells and related (frequently more detailed) data.

There can be some interplay between business rules and drill rules. For example, you might have a Cube A that uses business rules to pull values from a separate Cube B. You might then create a drill rule (and an accompanying drill process) that lets users drill-through from Cube A to Cube B to review the source values in full context.

Similarly, if the values in a cube were imported from an ODBC source, you could create a drill rule and process that lets users drill-through from the cube to a grid-formatted view of the ODBC source.

A drill rule uses the syntax `['`*`area_definition`*`']=S:'`*`drill_process_name`*`';`. *area_definition* is the section of the cube in which you want to enable drill-through. *drill_process_name* is the name of the TurboIntegrator process that is executed when you execute a drill-through from a cell that is included in the *area_definition*. You can learn more about setting up drill processes in "Create a drill process" on page 237.

Procedure

1. In edit mode, go to the database in the Data tree where you want to create a drill rule and expand the database to reveal the cubes.
2. Right-click the cube for which you want to create drill rules.
3. Click **Drill**, **Create drill rules**.

   The drill rules editor opens. This editor is identical to the business rules editor, but the icon on the upper left of the editor can help you distinguish between the two.

   - Drill rules editor icon ▦
   - Business rules editor icon ⊞

   For each cube area you want to associate with detailed data, continue with steps 8 - 11.
4. Enter the area definition for the region of the cube where you want to enable drill-through.

   The area definition can be as broad as the entire cube or as specific as a single cell.

   To specify the entire cube, enter an empty bracket set `[ ]`.

   To specify a single member, enter the member name enclosed in single quotation marks within brackets. For example, `['Price']`.

   To specify an area identified by multiple members, enter the members within brackets. Enclose each member name in single quotes and use a comma to separate member names. For example, the area definition `['Price','France','Jan']` applies to any cell identified by the members Price, France, and Jan.

   Tip: You can use the auto-complete or drag-and-drop features of the rules editor to simplify area definition. For more information, see "Use the rules editor" on page 229.
5. After you enter the area definition, type `=S:` in the rules editor.
6. After the colon, enter the name of the drill process that will be run when a user initiates drill-through from a cell within the area definition. Enclose the drill process name in single quotation marks.

   Important: When you create a drill process, Planning Analytics creates a control process with the `}Drill_` prefix. For example, when you create a drill process named `PriceCubeToSource`, Planning Analytics creates the `}Drill_PriceCubeToSource` control process. Do not include the `}Drill_` prefix when you specify a drill process name in a drill rule.

   You can associate more than one drill process with an area. Enclose all drill processes, separated by commas, within a set of single quotation marks.
7. Enter a semi-colon at the end of the rule to indicate the conclusion of the drill rule statement.
8. Click 💾.

Results

When a drill rule and associated drill processes are in place, you can right-click a cell that is included in the rule area definition and view underlying data. If the area is associated with a single drill process, the process is immediately run. The underlying data is displayed in either an exploration view for a cube view source, or a grid for an ODBC source.

If you initiate drill-through from a cell that is associated with multiple drill processes, you can choose which process you want to run.

Example

`[ ]=S:'PriceDrill';` this drill rule runs the `PriceDrill` drill process when you initiate drill-through from any cell in the cube.

`['Price']=S:'ODBCSourceDrill';` this drill rule runs the `ODBCSourceDrill` drill process when you initiate drill-through from any cell in the cube that is identified by the `Price` member.

`['Jan']=S:'PriceDrill, ODBCSourceDrill';` this drill rule lets you choose to run either the `PriceDrill` or `ODBCSourceDrill` drill process when you initiate drill-through from any cell in the cube that is identified by the `Jan` member.

`['Price','France','Jan']=S:'PriceDrill';` this drill rule runs the `PriceDrill` drill process when you initiate drill-through from any cell in the cube that is identified by the `Price`, `France`, and `Jan` members.

Edit a drill rule

Planning Analytics Workspace modelers and administrators can edit drill rules.

Procedure

1. In edit mode, go to the database in the Data tree where you want to edit a drill rule, and expand the database to reveal the cubes.
2. Right-click the cube for which you want to edit drill rules, and click **Drill**, **Edit drill rules**.

   Use the drill rules editor to modify your drill rules.

Delete a drill rule

Planning Analytics Workspace modelers and administrators can delete drill rules.

Procedure

1. Log in to Planning Analytics Workspace with a user that has the Modeler or Administrator role.
2. Make sure that you are in edit mode.
3. In the Data tree, navigate to the database where you want to delete a drill rule, and expand the **Cubes** group.
4. Right-click the cube for which you want to delete drill rules.
5. Click **Drill**, **Delete drill rules**, and click **OK**.

Create a drill process

A drill process is a special type of TurboIntegrator process that is called from a drill rule to display underlying data associated with a view cell. A drill process is run when you initiate drill-through from an exploration view.

About this task

A drill process can open a cube view or a DB connection (ODBC) source.

Procedure

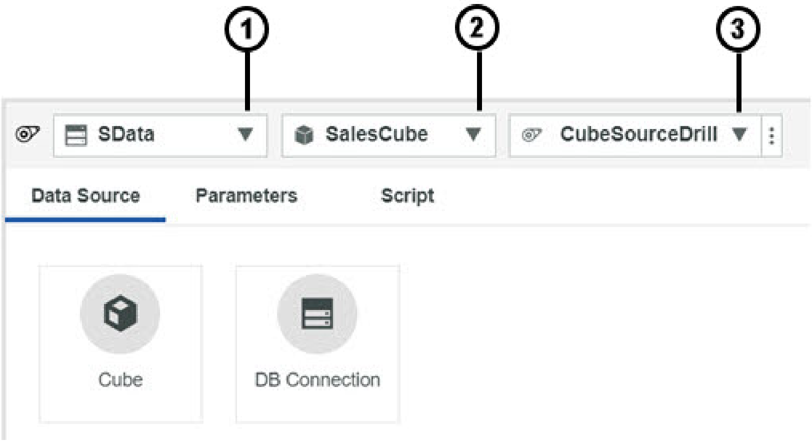
1. In edit mode, go to the database in the Data tree where you want to create a drill process, and expand the database to reveal the cubes.
2. Right-click the cube for which you want to create a drill process.

   This cube, from which a drill-through originates, is called the origination cube.
3. Click **Drill**, **Create drill process**.
4. Enter a name for the process, then click **Create**.

   Tip: For best practice, use a drill process name that identifies the origination cube for the drill process. For instance, if you create a process to drill from a cube that is named PriceCube to an ODBC source, you might name the drill process PriceCubeToODBCSource. This naming convention makes it easier to identify a drill process when you initiate drill-through from a cube view and must select from multiple drill processes that are associated with the cube.

The drill process editor opens. This editor is nearly identical to the regular process editor, but has only two data source options: **Cube** and **DB Connection**. These data sources are the only ones that are supported for drill-through.

Additionally, the drill process editor includes a cube selector, which is not part of the regular process editor. Drill processes are associated with a specific cube. If you pick a new cube from the cube selector, the process selector contains any drill processes associated with the selected cube.



1
: Database selector.

2
: Cube selector.

3
: Drill process selector.

5. To define a drill process with a Cube source, complete these steps:
   a) Click **Cube**.
   b) Select the cube that contains the view that you want to open, then click **Next**.

      You can open any cube that resides on the same database as the current drill process.
   c) Select the view that you want to open with the drill process. If such a view does not exist, click **Create view** to define a new view.
   d) Click **Load Preview**.
   e) Click **Validate**.
      If validation fails, see Troubleshooting.
   f) Click **Save**.

      TM1 saves the drill process as a TurboIntegrator process, but prefixes the name you assigned in step 8 with the string }Drill_. For example, if you save a drill process with the name PriceCubeToODBCSource, TM1 saves the process as }Drill_PriceCubeToODBCSource. Drill processes appear under the Control Objects for your database on the Planning Analytics Workspace content tree.
6. To define a drill process with a DB Connection source, complete these steps:

   Note: A Secure Gateway connection for the current Planning Analytics database must be in place before you can see any DB Connection sources in the drill process editor. See Administer IBM Secure Gateway for details on establishing a Secure Gateway.

   a) Click **DB Connection**.
   b) Select the connection that you want to use. Optionally, click **Log in credentials** and supply a username and password for the connection.
   c) Click **Next**.

d) Enter an SQL query to extract data from the source. The syntax and format of the query varies depending on which type of database you are accessing. If the query references a table name that contains spaces, you must enclose the name in double quotation marks.

e) Click **Test and preview**.

The process editor displays the first 10 records from the data source.

If the query doesn't return the results that you want, you can modify the query and click **Test query** to refine the results.

f) Click **Validate**.

If validation fails, see Troubleshooting.

g) Click **Save**.

TM1 saves the drill process as a TurboIntegrator process, but prefixes the name you assigned in step 8 with the string }Drill_. For example, if you save a drill process with the name PriceCubeToODBCSource, TM1 saves the process as }Drill_PriceCubeToODBCSource. Drill processes appear under the Control Objects for your database on the Planning Analytics Workspace content tree.

Results

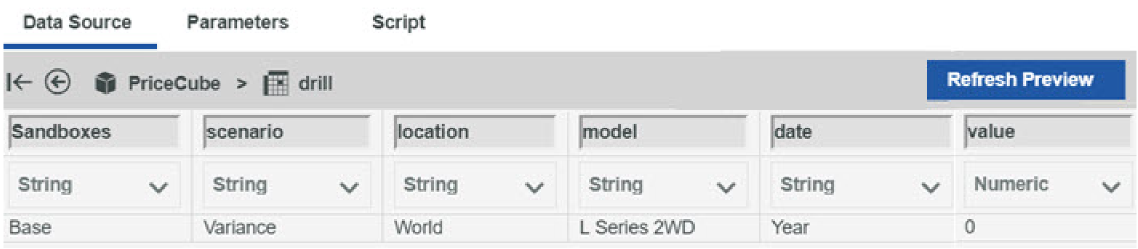
The drill process is now available to be included in a drill rule.

Troubleshooting

If any of the variables in your process have the same name as any of the parameters for the process, process validation fails with an Invalid Variable error.

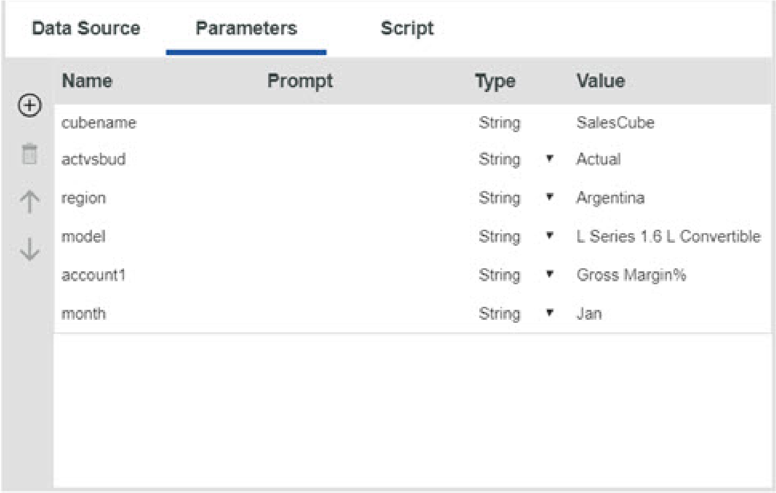
A drill process creates variables and parameters for each dimension in the origination cube.

Variable names, which are created based on the content of your data source, appear in each column on the **Data Source** tab. In the following image, sandboxes, scenario, location, model, date, and value are all variables.



Sandboxes	scenario	location	model	date	value
String	String	String	String	String	Numeric
Base	Variance	World	L Series 2WD	Year	0

Parameter names appear in the Name column on the Parameters tab.



Name	Prompt	Type	Value
cubename		String	SalesCube
actvsbud		String ▾	Actual
region		String ▾	Argentina
model		String ▾	L Series 1.6 L Convertible
account1		String ▾	Gross Margin%
month		String ▾	Jan

Parameter names are derived from the dimension names in the origination cube.

If validation fails due to identical naming, review the variable names and parameter names for your process. Modify any variable names that are identical to parameter names, then click **Validate** again.

Edit a drill process

Planning Analytics Workspace modelers and administrators can edit drill processes.

Procedure

1. In edit mode, go to the database in the Data tree where you want to edit a drill process, and expand the database to reveal the cubes.
2. Right-click the cube for which you want to edit a drill process.
3. Click **Drill**, **Edit drill process**.
4. Select the drill process, then click **Edit**.

   Use the drill process editor to modify the process.

Delete a drill process

Planning Analytics Workspace modelers and administrators can delete drill processes.

Procedure

1. In edit mode, go to the database in the Data tree where you want to delete a drill process, and expand the database to reveal the cubes.
2. Right-click the cube for which you want to delete a drill process.
3. Click **Drill**, **Delete drill process**.
4. Select the drill process, then click **Delete**.

Object security

Modelers manage the security for cubes, dimensions, and processes in the settings editor, and security for dimension members in the security view in the dimension editor. You can assign security to these objects for any non-administrative user group in Planning Analytics Workspace.

You can also secure cells. Cell level security overrides all other Planning Analytics Workspace security.

To configure security, do the following steps:

1. Create user groups in the dimension editor.
2. Add users to the groups. Users are added to the groups in the **}ClientGroups** cube, which is in the **Control Objects** node of the content tree. You can also use TurboIntegrator processes to add users to groups, see AssignClientToGroup.
3. Define the access for cubes, dimensions, and processes for each group in the settings editor.
4. Define the access for dimension members in the dimension editor.
5. Define the access to cells if required.

Users are invited to IBM Planning Analytics Workspace by a Planning Analytics administrator.

To find out more, see “Upload a file to add or remove multiple users (cloud only)” on page 274.

Related topics:

Managing Users and Groups

IBM Business Analytics Proven Practices: How to implement element or cube-based security for IBM Cognos TM1

Create a user group

You manage security in Planning Analytics Workspace by organizing users into groups.

Two types of user groups exist.

Administrative Groups
Predefined groups of ADMIN, DataAdmin, SecurityAdmin, and OperationsAdmin. You can't modify these groups. For more information, see the TM1 Operations guide, Administrative groups and authority.

User Groups
Includes all user-created groups.

You create user groups in the **Settings editor** or in the security view 🔒 in the dimension editor. You can also create user groups by using the process ADDGROUP.

Procedure

1. In edit mode, go to the database in the Data tree where you want to create a user group and expand the cubes.
2. Right-click **Cubes** and click **Edit settings**
3. To create a user group, click ⊕.
   The **Create New User Group** dialog box opens.
4. Enter the group name and click **Create**.

What to do next

Next, add users to the user groups. Then, define the access for the user groups to dimension members, cubes, dimensions, processes, chores, and cells as needed.

Add users to user groups

Modelers can add users to user groups in the **}ClientGroups** cube.

Before users can be added to user groups, they must first be added to Planning Analytics Workspace by an administrator. To find out more, see "Upload a file to add or remove multiple users (cloud only)" on page 274.

Tip: You can also use processes to add users to user groups, with the AddClient and AssignClientToGroup processes.

Procedure

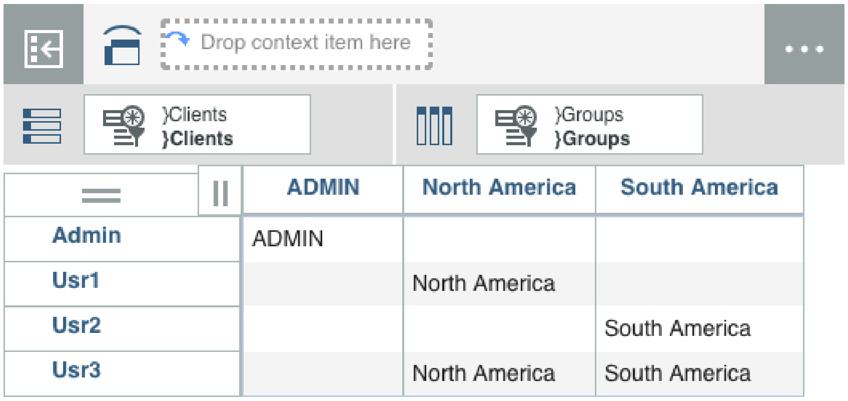
1. In the Data tree, expand the appropriate database so that you can see **Control Objects**, and navigate to **Control Objects** > **Cubes** > **}ClientGroups**.
2. Right-click **}ClientGroups** and select **Add new view**.
   The cube has two dimensions: **}Clients**, which contains all the users, and **}Groups**, which contains all the user groups.
3. Find the user in the **}Clients** dimension, and the user group that you want to add the user to in the **}Groups** dimension and click the cell at the intersection.
4. Type the name of the user group in the cell.

Example

The SData sample database contains some predefined groups and users.

User name	Group membership
Admin	ADMIN
Usr1	North America
Usr2	South America
Usr3	North America and South America

The following image shows the **}ClientGroups** cube view in the SData sample database.



	ADMIN	North America	South America
Admin	ADMIN		
Usr1		North America	
Usr2			South America
Usr3		North America	South America

Secure dimension members

You manage security for dimension members in the dimension editor.

By default, all groups that have access to the database have write access to the dimension members. The first time that you open the security pane, a message is displayed: `Member security is presently disabled.` If you enable security for dimension members, the default access is removed, and you must set the permissions to whatever access that the user groups require. If new members are later added to a dimension, assign access permissions to the appropriate user groups.

To find out more about what happens when you enable security, see “ The security cube” on page 243.

Procedure

1. Open the dimension edit for whose members you are setting the security for, and click 🔒 to open the security pane.
2. Click **Enable**.
3. For each cell, type or select the security level. You can select a range of contiguous or non-contiguous cells, and right-click to select the security. To set security for columns or rows, select and right-click the row or column headers, click **Set security**, and select an option.

   The security levels are as follows:

   NONE
   No objects can be seen.

   READ
   Can view objects, but cannot perform operations on the objects.

   WRITE
   Can view and update objects.

   RESERVE
   Can view and edit objects, and can temporarily reserve objects to prevent other users from updating them.

LOCK

Can view and edit objects, and can permanently lock objects to prevent all users (including the person who set the security) from updating them.

ADMIN

Complete access to objects.

Example

Say that you have a Regions dimension with the following members:

- Total
  - Americas
    - Canada
    - United States
  - Europe
    - France
    - Germany
    - United Kingdom
    - Switzerland

You might have two user groups, Americas_group, Europe_group, and assign functions as follows:

Americas_group

Write access to Americas.

Read access to Europe.

Europe_group

Write access to Europe.

Read access to Americas.

The security cube

Security settings for dimension members are stored in a security cube called }ElementSecurity_*dimension_name* cube.

The }ElementSecurity_*dimension_name* cube does not exist by default, and all users have access to the dimension members by default.

When you edit the security setting for a dimension for the first time in the dimension editor, or create a user group, you are prompted to create a security cube.

When the security cube is created, users must be given explicit write access to all of the members in the dimension.

The security cube contains the following dimensions:

Dimension_name

The dimension and members for which the security cube is created for.

}Groups

User groups that are defined for the TM1 database.

The }ElementSecurity_*dimension_name* cube is in the **Control Objects** > **Cubes** node of the content tree.

Secure cubes, dimensions, processes, and chores

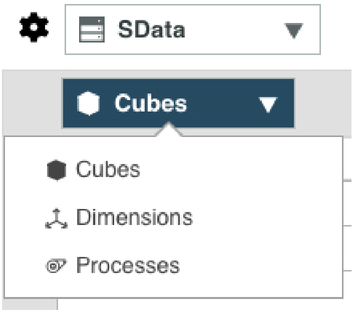
Modelers can manage security settings for cubes, dimensions, processes, and chores in the settings editor.

Note: In earlier versions of Planning Analytics Workspace, if you gave access to a cube, you also had to set access to at least READ for its component dimensions. From version 2.0.44, when a security group is given access to a cube, READ access is automatically given to its dimensions. Also, when security access is removed from a dimension, the security access of all the cubes that contain the dimension is set to none.

Procedure

1. Find the database that you want to manage security for in the tree, and right-click either **Cubes**, **Dimensions**, **Processes**, or **Chores**.
2. Click **Edit settings**.
3. Click 🔒 to open the security editor.

   The security editor has a view with either dimensions, cubes, processes, or chores on the rows, and user groups on the columns. Switch between dimensions, cubes, processes, or chores by using the menu.

4. For each cell, type or select the security level. You can select a range of contiguous or non-contiguous cells, and right-click to select the security. To set security for columns or rows, select and right-click the row or column headers, click **Set security**, and select an option.

   Security levels are shown. Note, only NONE and READ options are available for processes.

   NONE
   No objects can be seen. When NONE is selected, the cell is blank in the editor.

   READ
   Can view objects, but cannot write to the objects.

   WRITE
   Can view and update objects.

   RESERVE
   Can view and edit objects, and can temporarily reserve objects to prevent other users from updating them.

   LOCK
   Can view and edit objects, and can permanently lock objects to prevent all users (including the person who set the security) from updating them.

   ADMIN
   Complete access to objects.

5. You can find or filter on the items in the row headers.

   - To highlight all instances of a search phrase, click 🔍, type the search phrase in the field and press Enter. The first member in the dimension that contains the phrase is highlighted. The total number of members that are found is displayed, and you can cycle through the members by clicking the up and down arrows 1 of 10 ^ ˅ in the **Find Members** field.
   - To filter on a search phrase, click ▽ type the search phrase in the field and press Enter.

Secure cells

You can secure specific cells in Planning Analytics Workspace.

Cell-level security overrides all other security. If you want member-level security to apply, cell-level security for the cell must be undefined.

When you secure cells, a cell security control cube is created. The cell security control cube contains all of the dimensions of the cube that you apply cell-level security to. Then, set the security for cells in the control cube by assigning security permissions to user groups.

Cell-level security applies to leaf members and generally does not apply to consolidations, although you can use the andNone Read security permissions to control the display or editing of consolidations.

Before you begin

The user group that you are assigning access to must have access to the cube and the dimensions in the cube. See "Secure cubes, dimensions, processes, and chores" on page 243 to find out more.

Procedure

1. In the Data tree, right-click the cube and click **Edit cell security**.

   A cell security view, and the rule editor are added to the sheet. The cell security cube is created in **Control Objects** > **Cubes** and is called **}CellSecurity_*cubename***. You can create rules to assign cell security, or you can assign cell security directly in the cell security view.

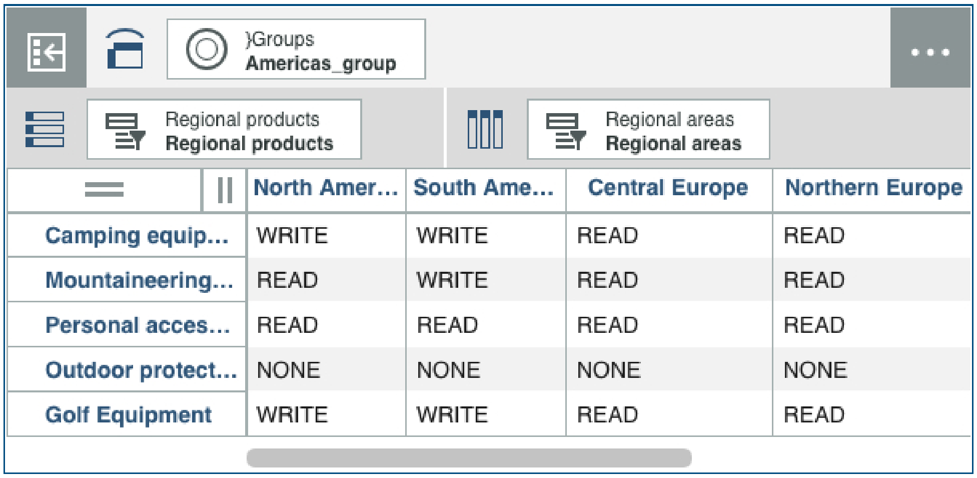
2. To assign cell security in the rule editor, type the rule in the rule editor.

```
['member','security_group'] = S:'security_level'
```

   The following example prevents any users in Europe_group from viewing any cells that are identified by the members pineapple and France.

```
['pineapple','France','Europe_group'] = S:'NONE';
```

3. To assign cell security in the cell security view, in the view, drag the **}Groups** view onto the context and select the user group that you want to assign security to.



}Groups
Americas_group

Regional products
Regional products

Regional areas
Regional areas

	North Amer...	South Ame...	Central Europe	Northern Europe
Camping equip...	WRITE	WRITE	READ	READ
Mountaineering...	READ	WRITE	READ	READ
Personal acces...	READ	READ	READ	READ
Outdoor protect...	NONE	NONE	NONE	NONE
Golf Equipment	WRITE	WRITE	READ	READ

4. Type the permissions in the cells that you are applying access to.

NONE
No objects can be seen. When NONE is selected, the cell is blank in the editor.

READ

Can view objects, but cannot write to the objects.

WRITE
Can view and update objects.

RESERVE

Can view and edit objects, and can temporarily reserve objects to prevent other users from updating them.

LOCK

Can view and edit objects, and can permanently lock objects to prevent all users (including the person who set the security) from updating them.

ADMIN

Complete access to objects.

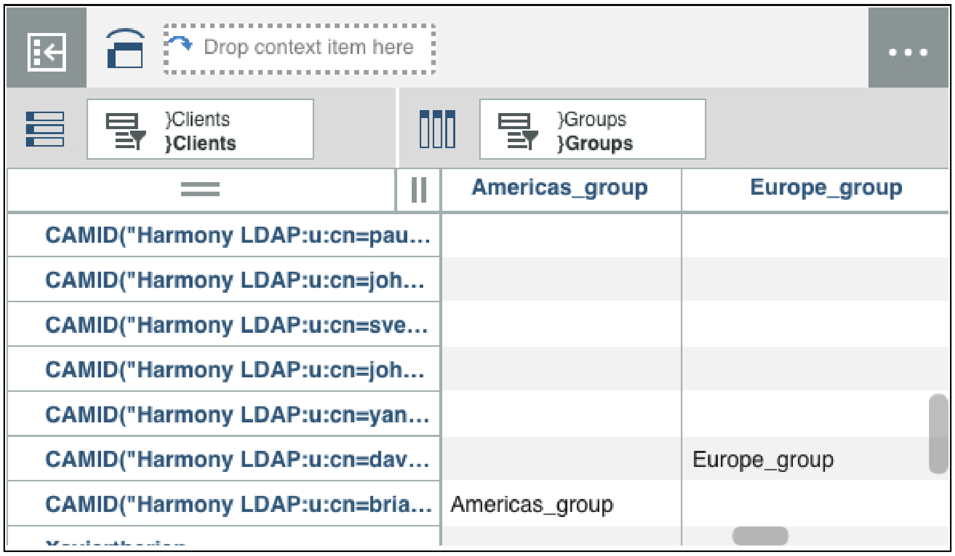
Cell security example

This example starts with a simple Products cube with a regional areas dimension and regional products dimension.

Some product ranges are only sold in some regions, so you want to prevent data from being entered for these regions.

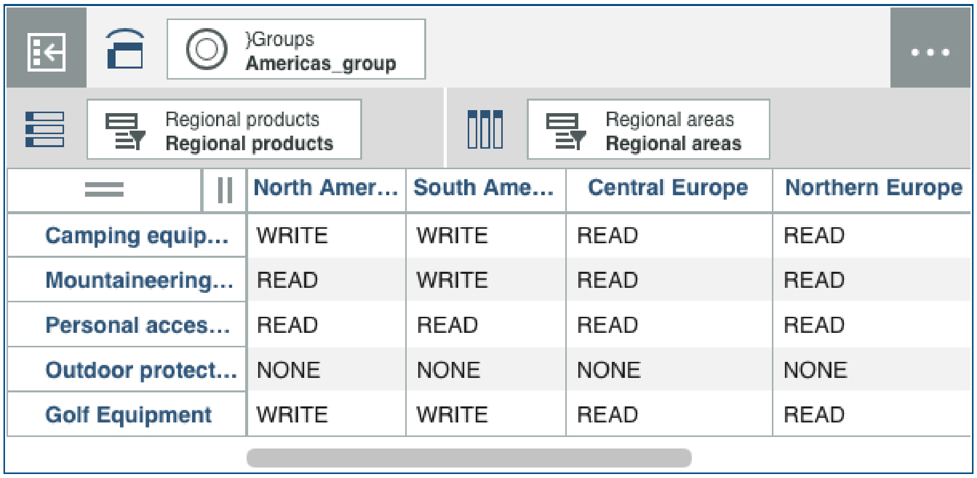
The steps to prevent data from being entered for product ranges in some regions are as follows.

1. You are logged on to Planning Analytics Workspace as an administrator. Create a user group in the Security tab of the dimension editor, and name it **Americas_group**.
2. In the **}ClientGroups** cube, add the sales manager for the Americas territory to the **Americas_group**.



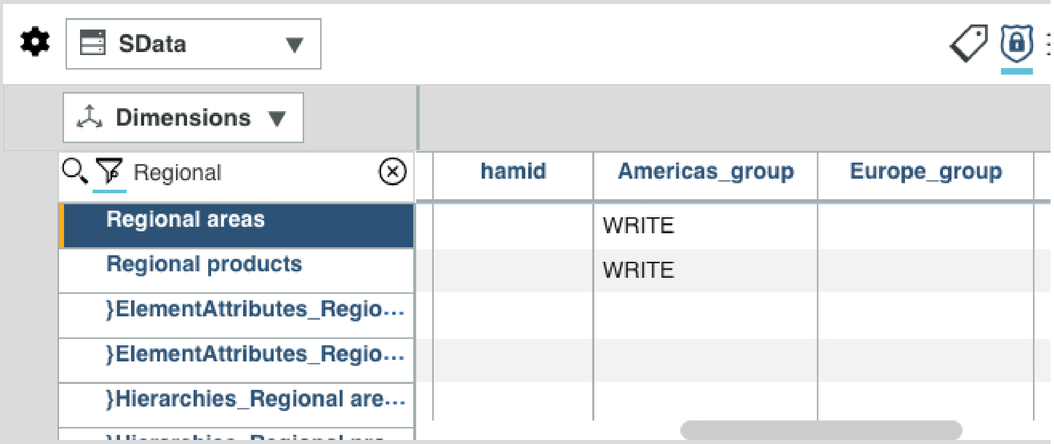
	Americas_group	Europe_group
CAMID("Harmony LDAP:u:cn=pau...		
CAMID("Harmony LDAP:u:cn=joh...		
CAMID("Harmony LDAP:u:cn=sve...		
CAMID("Harmony LDAP:u:cn=joh...		
CAMID("Harmony LDAP:u:cn=yan...		
CAMID("Harmony LDAP:u:cn=dav...		Europe_group
CAMID("Harmony LDAP:u:cn=bria...	Americas_group	

3. In the Data tree, right-click the Regional products cube, and select **Edit cell security** . Both the rules editor and the cell security views are opened on the sheet. You can either set up the security by using rules, or you can set the security in the cell security view. This example uses the cell security view.
4. In the cell security view, drag the **}Groups** dimension onto the context in the overview and find the **Americas_group**. Assign some different security to the cells for the **Americas_group** so that you can verify that the security is set up correctly. For example, a mixture of WRITE, READ, and NONE.



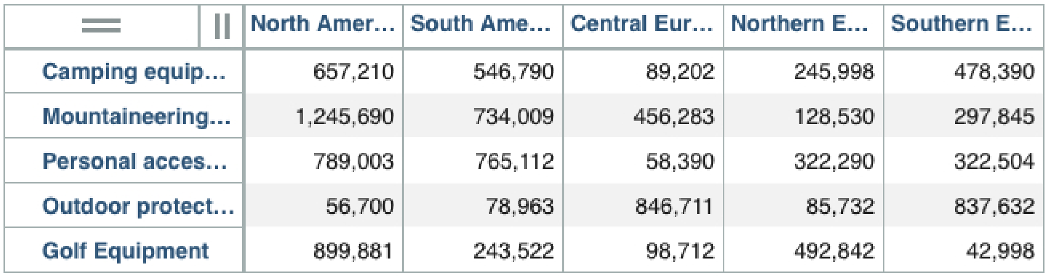
	North Amer...	South Ame...	Central Europe	Northern Europe
Camping equip...	WRITE	WRITE	READ	READ
Mountaineering...	READ	WRITE	READ	READ
Personal acces...	READ	READ	READ	READ
Outdoor protect...	NONE	NONE	NONE	NONE
Golf Equipment	WRITE	WRITE	READ	READ

5. In the Data tree, go to the database and right-click **Cubes** > **Edit settings** to open the Settings editor and give WRITE access to the Regional products cube for **Americas_group**. Then, select **Dimensions** from the **Settings editor** menu, and give WRITE access to the Regional areas and Regional products dimension for the **Americas_group**.



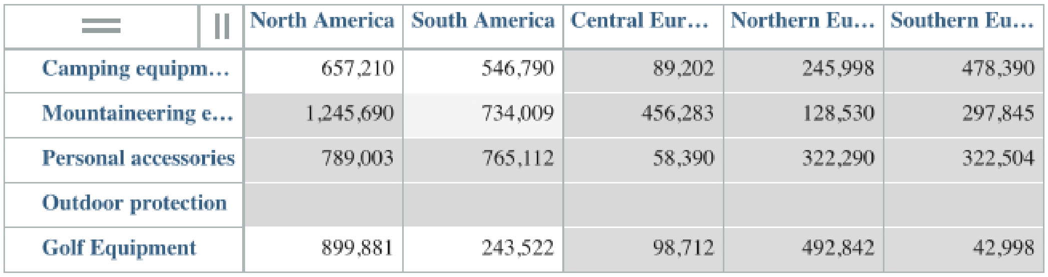
	hamid	Americas_group	Europe_group
Regional areas		WRITE	
Regional products		WRITE	
}ElementAttributes_Regio...			
}ElementAttributes_Regio...			
}Hierarchies_Regional are...			

6. Add the Regional products cube view to the sheet and save the view as a separate object. Click the view to display the toolbar, click **Save** , then select **Save as** to specify a name.
7. Enter some numbers into the view.



	North Amer...	South Ame...	Central Eur...	Northern E...	Southern E...
Camping equip...	657,210	546,790	89,202	245,998	478,390
Mountaineering...	1,245,690	734,009	456,283	128,530	297,845
Personal acces...	789,003	765,112	58,390	322,290	322,504
Outdoor protect...	56,700	78,963	846,711	85,732	837,632
Golf Equipment	899,881	243,522	98,712	492,842	42,998

8. Open a different browser, and log on as the Americas sales manager who is a member of the **Americas_group**, and is not a member of any other group.
9. Open the Regional products view.



	North America	South America	Central Eur…	Northern Eu…	Southern Eu…
Camping equipm…	657,210	546,790	89,202	245,998	478,390
Mountaineering e…	1,245,690	734,009	456,283	128,530	297,845
Personal accessories	789,003	765,112	58,390	322,290	322,504
Outdoor protection					
Golf Equipment	899,881	243,522	98,712	492,842	42,998

In the example shown, you can see that the Americas sales manager can see nothing for Outdoor protection products. Other cells shaded dark gray are readable but not writable, and cells with a white or light gray background are writable.

Delete a user group

You can delete user groups in the security view in the **Dimension Editor** when you create or edit members of a hierarchy.

Procedure

1. To open the security pane, in the **Dimension Editor**, click 🔒.
2. To delete a user group, select the user group and click 🗑.
3. Click **Ok** to confirm that you want to remove the selected user group.

TurboIntegrator processes

IBM Planning Analytics Workspace provides the ability to create, edit, and execute TurboIntegrator processes.

You can use TurboIntegrator processes to import or export data, create and maintain metadata/objects, and manage security on your database.

You can use the Planning Analytics Workspace process editor to edit processes that were created in other TM1 clients, such as Architect or Perspectives.

For complete details on TurboIntegrator processes, see the TM1 TurboIntegrator documentation. This documentation references a different user interface, which includes some capabilities that are not present in Planning Analytics Workspace. However, the general concepts, descriptions of process procedures, and examples are valid for both environments.

Create and edit processes

You can create or edit a TurboIntegrator process in the process editor.

Procedure

1. To create a new process:
   a) In the Data tree, expand the database on which you want to create the new process.
   b) Right-click the **Processes** group.
   c) Click **Create process**.
   d) You can change the database where the process is stored by selecting a database from the list. The current database is selected by default.
   e) Enter a name for the process.
   f) Click **Create**.
2. To edit an existing process:

a) Right-click the process in the Data tree.

b) Click **Edit process**.

3. Enter your process statements directly in the process editor. Process statements are generally TM1 TurboIntegrator or Rules functions that execute actions upon data or metadata in your database.

   - For full descriptions of all TurboIntegrator functions, see TM1 TurboIntegrator Functions on IBM Knowledge Center.
   - For full descriptions of all Rules functions, see TM1 Rules Functions on IBM Knowledge Center. All Rules functions, with the exceptions of STET and ISLEAF, are valid in TurboIntegrator processes.

   A TurboIntegrator process has four distinct procedures that are executed sequentially when you run a process.

Procedure	Description
Prolog	A series of statements to be executed before the data source is processed.
Metadata	A series of statements that update or create cube, dimensions, and other metadata structures during processing.
Data	A series of statements that manipulate values for each record in the data source.
Epilog	A series of statements to be executed after the data source is processed.

   You must ensure that you create statements in the appropriate procedure within the process. For example, a statement that creates a new cube should be in the Metadata procedure. Similarly, any statements that update data values should be in the Data procedure.

   When you enter a statement within any procedure section in the process editor, it must be either before this line:

   ```
   #****GENERATED STATEMENTS START****
   ```

   or after this line:

   ```
   #****GENERATED STATEMENTS FINISH****
   ```

   Do not insert any of your statements between the generated statements start and finish lines, as anything between these lines can be overwritten.
4. If you define parameters for the process, you can click **Script** to return to the script view.
5. Click **Save** to save the process.
6. Alternatively, click ⋮ and then click **Save as** to save as a different name.

Define a data source

You can use the Planning Analytics Workspace process editor to import data from multiple data source types.

You can import from a cube view or a dimension subset. Each data source type requires specific steps to identify and define the data source.

Define a cube view data source

The Planning Analytics Workspace process editor lets you define a cube view as a data source from which you can extract data and create or update objects and/or data.

Procedure

1. Click the **Data Source** tab on the process editor.
2. Click **Cube**.
3. Click the cube that contains the view you want to use as a data source, then click **Next**.
4. Select the view you want to use as the data source.

   If the view you want to use does not already exist, click **Create view** to define a new view.

   a) Type the **New view name**.

   b) Set the skip options:

   Skip zeros
   Turn this option on to ignore zeros or blank values when extracting the view. Turn this option off to include zeros or blank values when extracting the view. The default is on.

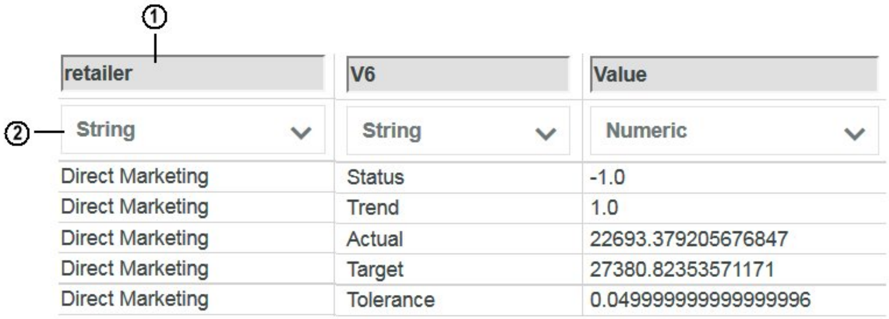
   Skip consolidations
   Turn this option on to ignore values derived through consolidation when extracting the view. Turn this option off to include values derived through consolidation when extracting the view. The default is on.

   Skip rule values
   Turn this option on to ignore values derived through rules when extracting the view. Turn this option off to include values derived through rules when extracting the view. The default is on.

   c) For each dimension in the source cube, pick the hierarchy and the set you want to use in the view.

   d) Click **Create**.
5. Select the new view that you just created.
6. Click **Load preview**.
7. Click **Load Preview**.

   The process editor loads several records from the cube view to help you identify the structure of the source data. Each column in the source is assigned a variable name. The data type for each column is identified as either string or numeric.



retailer	V6	Value
String	String	Numeric
Direct Marketing	Status	-1.0
Direct Marketing	Trend	1.0
Direct Marketing	Actual	22693.379205676847
Direct Marketing	Target	27380.82353571171
Direct Marketing	Tolerance	0.049999999999999996

1
variable name

2
data type

Assigned variable names generally coincide with the dimension names in the data source, but can also be a combination of the letter V and a number, such as V1 or V6. If the variable names that are assigned are not meaningful or familiar to you, you can click a variable name and type a new name. Variable names can contain only letters, numbers, and underscore and must start with a letter.

It's good practice to give the variables a meaningful name. Having meaningful names makes the process scripts easier to read and troubleshoot.

Similarly, if the data type for any column is misidentified, you can click the data type and pick a different type.

Define a dimension set data source

Follow these steps to define a dimension set as a data source for a process.

Procedure

1. Click the **Data Source** tab on the process editor.
2. Click **Dimension**.
3. Select the dimension that contains the set you want to use as a data source, then click **Next**.
4. Select the set you want to use as the data source.

   If there is no existing set that is appropriate for use as a data source, click **Create set**.

   a) Use the set editor to create the set you want to use as a data source. See "Create and edit sets" on page 141 for details on using the set editor.

   b) After you've defined the set, click **Save** on the set editor.

   c) On the **Save set as** screen, enter a name for the set.

   d) Enable the **Share public** option.

   Important: The **Share public** option must be enabled to make the new set available as a data source for the process. Private sets cannot be used as process data sources.

   e) Click **Save** on the **Save set as** screen.

   f) Close the set editor.

   g) Pick the set that you just created as the data source.
5. Click **Load Preview**.

   The process editor loads several members from the set to help you identify the structure of the data source. The data from the set is assigned a variable name, which generally coincides with the name of the dimension that contains the source data. The data type for the data is identified as either string or numeric.

   If the variable name that is assigned is not meaningful to you, you can click a variable name and type a new name. Variable names can contain only letters, numbers, and underscore and must start with a letter.

   Similarly, if the data type is misidentified, you can click the data type and pick a different type.
6. Click **Save**.

Define a file data source

You can define a text file as a data source for a process.

About this task

The file you use as a data source must be a delimited text file. It can have any file extension.

Microsoft Excel (.xls/.xlsx) files cannot be used as a data source, nor can fixed-width text files.

Procedure

1. Click **Data Source** on the process editor.
2. Click **File**.
3. Click the file that you want to use as a data source.

The **File** menu lists the files that have previously been used as process data sources on your TM1 database.

If the file you want to use as a data source is not available in the list, drag and drop the new source file onto the **Drag and drop file here** region of the process editor. You can also click on the **Drag and drop file here** region to browse to the file you want to use as a data source.

The file is uploaded to the TM1 database and is set as the process data source. It also becomes available as a data source for future processes.

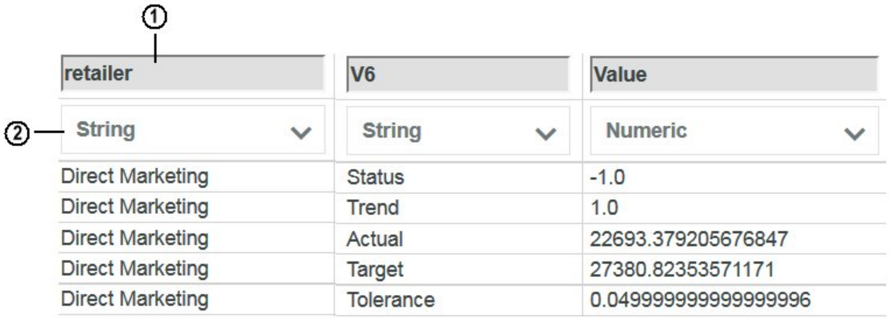
4. Click **Next**.

   The process editor identifies the delimiter character, as well as the quote character, decimal separator, and thousand separator, and number of header records in your source. If the process editor incorrectly identifies any of these items, you can modify the settings.

   Important: Your data source file might contain one header record, multiple header records, or zero header records. You need to know how many header records your source contains and set the **Header records** accordingly to ensure that all data in the source file is processed correctly.

5. Click **Load Preview**.

   The preview region of the process editor reveals the structure of your source file.



retailer	V6	Value
String	String	Numeric
Direct Marketing	Status	-1.0
Direct Marketing	Trend	1.0
Direct Marketing	Actual	22693.379205676847
Direct Marketing	Target	27380.82353571171
Direct Marketing	Tolerance	0.049999999999999996

1

variable name

2

data type

If your source file includes one or more header records, the assigned names for string variables generally coincide with the values in the header records. but can also be a combination of the letter V and a number, such as V1 or V6. Assigned numeric variable names use the V1 convention. If the variable names that are assigned are not familiar to you, you can click a variable name and type a new name. It's good practice to give the variables a meaningful name; this makes the process scripts easier to read and troubleshoot. Variable names can contain only letters, numbers, and underscore and must start with a letter.

Similarly, if the data type for any column is misidentified, you can click the data type and pick a different type.

6. Click **Save**.

Define a database connection data source

You can define a database connection as a data source for a process.

About this task

To use a database connection as a data source for a process, the following conditions must be met:

- For both Planning Analytics Workspace on cloud and Planning Analytics Workspace Local, you must be IBM Planning Analytics version 2.0.4 or later.

- For Planning Analytics Workspace on cloud, if the database resides anywhere other than the computer on which your TM1 database resides, you must "Create a Secure Gateway" on page 335 to access the data.
- For Planning Analytics Workspace Local, the client software for your relational database must be running on the same computer on which your TM1 database resides. For example, an ODBC Excel data source cannot be located on a mapped network drive; it must exist on the computer where the TM1 database resides.
- Regardless whether you are working with Planning Analytics Workspace on cloud or Planning Analytics Workspace Local, the ODBC data source must already be established before you can use it as a process data source.

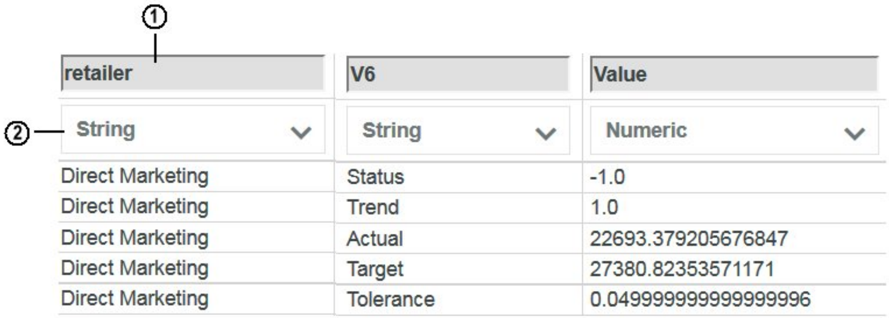
Procedure

1. Click **Data Source** on the process editor.
2. Click **DB Connection**.
3. Select the database connection you want to use as a process data source.
4. Optionally, click **Log in credentials** and provide a username and password to access the database connection.
5. Click **Next**.
6. Enter an SQL query to extract data from the source. The syntax and format of the query will vary depending on which type of database you are accessing.

   If the query references a table name that contains spaces, you must enclose the name in double quotes.

   Tip: While entering a query, you can press CTRL+space to auto-complete a keyword. If the cursor is preceded by a blank space, you can press CTRL+space to view a list of available SQL keywords.
7. Click **Test and preview**.

   The preview region of the process editor reveals the structure of your source data.



retailer	V6	Value
String	String	Numeric
Direct Marketing	Status	-1.0
Direct Marketing	Trend	1.0
Direct Marketing	Actual	22693.379205676847
Direct Marketing	Target	27380.82353571171
Direct Marketing	Tolerance	0.049999999999999996

   1
   variable name

   2
   data type

   The variable names that are assigned generally coincide with the column names in your source data, but can also be a combination of the letter V and a number, such as V1 or V6. If the variable names that are assigned are not familiar to you, you can click a variable name and type a new name. It's good practice to give the variables a meaningful name; this makes the process scripts easier to read and troubleshoot. Variable names can contain only letters, numbers, and underscore and must start with a letter.

   Similarly, if the data type for any column is misidentified, you can click the data type and pick a different type, or click **Ignore** to exclude the column from processing.

   Note: If the query returns unsatisfactory results, you can modify the query and click **Test query** to refresh the preview.

Allow a process to modify security data

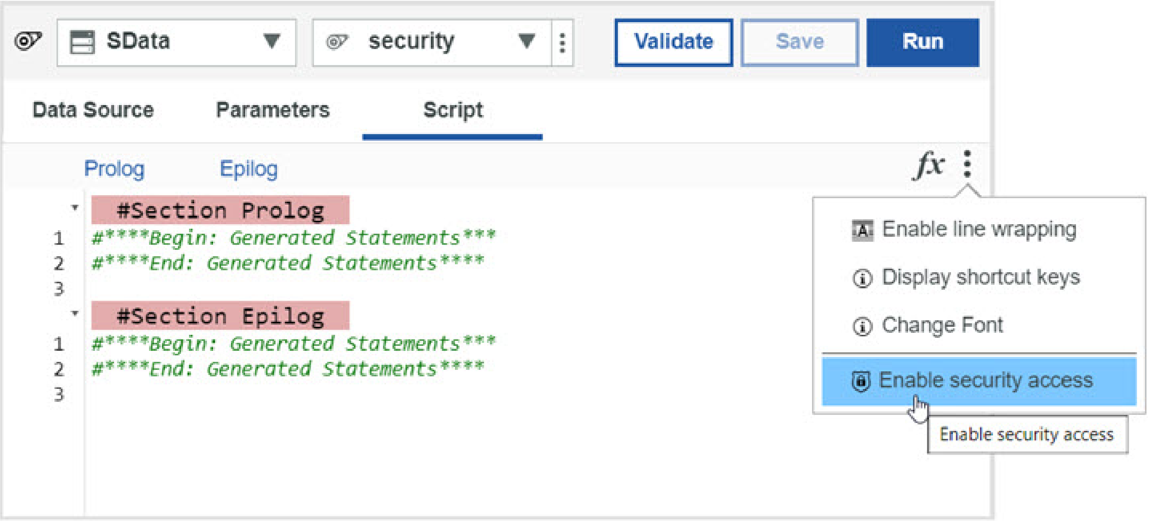
By default, security access is disabled when you create a new process. This means that the process cannot modify security data. If you want to allow a process to modify security, you must enable security access for the process.

About this task

You must have an Administrator or Modeler role in Planning Analytics Workspace and be a member of either the ADMIN or SecurityAdmin group in TM1 to enable security access. When security access is enabled, only Planning Analytics Workspace users who are members of TM1 ADMIN group can modify the process.

Procedure

1. Open the process editor.
2. Click the **Script** tab.
3. Click **Options**, then click **Enable security access**.



```
#Section Prolog
#****Begin: Generated Statements***
#****End: Generated Statements****

#Section Epilog
#****Begin: Generated Statements***
#****End: Generated Statements****
```

4. To disable security access after it has been enabled, click **Options**, then click **Disable security access**.

Create and edit process parameters

A TurboIntegrator process can include runtime parameters that are passed into the process upon execution. Parameter values are set when a user runs the process and answers prompts for parameter values.

Procedure

1. Click **Parameters** on the process editor to open the parameters view.
2. In the **Name** cell, enter a parameter name. Parameter names should adhere to "Naming conventions" on page 644, and should not include any TurboIntegrator reserved words.
3. In the **Prompt** cell, enter the question or prompt that a user will respond to when supplying a parameter value.
4. In the **Type** cell, choose a parameter type: numeric or string.
5. In the **Value** cell, enter a default value for the parameter.
6. Click ⊕ to create additional parameters, then repeat steps 2 through 5.
7. To change the order of parameters, select a parameter, then use the arrow buttons to move the parameter up or down in the process editor.
8. To delete a parameter, select the parameter, then click 🗑.
9. If necessary, click **Script** to return to the script view of the process editor.

Results

When a user runs the process, a dialog box requests answers to the prompts that you defined in step 3. The answers that the user supplies are passed into the process as parameter values.

Validate a process

You can validate a TurboIntegrator process to check for errors that are not identified by the syntax error highlighting feature in the process editor.

About this task

Validation applies to the current state of the process, not the most recently saved version of the process. You don't need to save a process before validating it.

Procedure

Click **Validate** on the process editor to validate the process.

Results

If the process is valid and free of errors, the message `Process <process_name> is valid` appears at the top of the process editor.

If there are errors in the process, an error message displays a brief description of the first error in the process. Where applicable, the message indicates the procedure and line number at which the error occurs. You should correct the error before attempting to validate the process again.

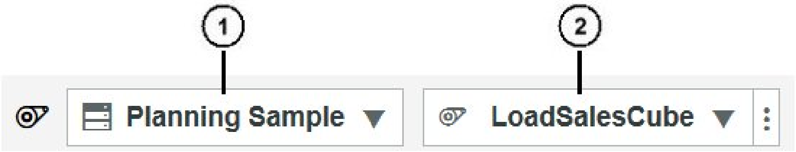
The error checking that occurs when you validate a process also occurs when you save a process. You can save a process with errors, but the errors must be resolved before you can successfully run the process.

Use the process editor

The Planning Analytics Workspace process editor includes several features that simplify the creation and validation of TurboIntegrator processes.

Process switching

You can use the **Database** and **Process** selectors to open a different process.



1
Database selector

2
Process selector

To open a new process that resides on the same database as the current process, click the **Process** selector and then click the process you want to open.

To open a process that resides on a different database, click the **Database** selector and pick the appropriate database, then click the **Process** selector and click the process you want to open.

Procedure links

A TurboIntegrator process contains four procedures: Prolog, Metadata, Data, and Epilog. The process editor includes links to each procedure so you can quickly jump to the procedure you want to edit.

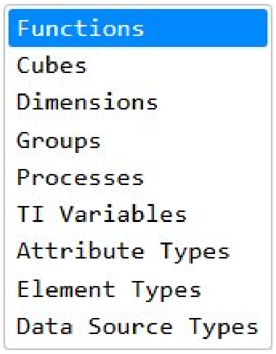
Procedure section collapse/expand

Click the small arrow next to any procedure section to collapse or expand the contents of the procedure.

Auto-completion

You can use the CTRL+SPACE keystroke combination to choose applicable items in the process editor.

- If you use CTRL+SPACE at the beginning of an empty line in the process editor, you'll see this list of categories:

  ```
  Functions
  Cubes
  Dimensions
  Groups
  Processes
  TI Variables
  Attribute Types
  Element Types
  Data Source Types
  ```

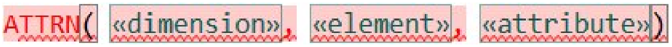
  Double-click any of the categories to see a list of specific of items, then double-click an individual item to insert it into the editor
- If you use CTRL+SPACE after typing one or more characters of a function name, a list presents all functions that start with the characters you've typed. You can then double-click the desired function to insert it into the editor.
- When the cursor is positioned immediately before a parameter placeholder in a function, use CTRL +SPACE to display a list of valid parameter values, or in the case when there is only a single valid parameter value, to insert the value. If the cursor is positioned after a comma following a parameter, CTRL+SPACE lets you pick another parameter value of the same type as the previous parameter. For example, if the cursor is positioned after comma following a dimension name in the CubeCreate function, you can pick another dimension name to insert into the function.

Function list

You can click *fx* on the Script tab of the process editor to quickly insert a TurboIntegrator function into the process.

When you click *fx*, you see a list of function categories. Click a function category to see the available functions within the category, then double-click a function to insert it at the cursor in the process editor.

The function is inserted with placeholders for required parameters. You must replace the placeholders with valid parameter values. For example, if you use the function list to insert the ATTRN function into your process, you'll see this:



```
ATTRN( «dimension», «element», «attribute»)
```

You must replace the <<dimension>>, <<element>>, and <<attribute>> placeholders to successfully return a numeric attribute value with ATTRN.

Shortcut keys

There are many shortcut keys available in the process editor. You can use these keys to edit statements, find and replace text, and navigate through the statements in the process editor.

To see a full listing of shortcut key combinations, click ⋮ on the Script tab of the process editor, then click **Display shortcut keys**.

To hide the listing of shortcut keys, click ⋮, then click **Hide shortcut keys**.

Syntax highlighting

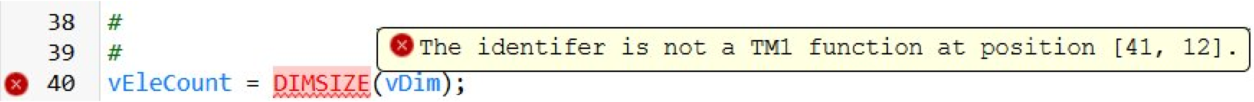
Syntax highlighting uses colors to let you quickly identify process components.

- comments: green
- local variables, declared within the process code: light blue
- variables created outside of the process code: dark blue
- reserved keywords, such as function names, conditions ('if' , 'while', 'endif'), or predefined variables: magenta
- strings: red
- parameter names, shown by the auto-complete feature: teal
- everything else: black

Error highlighting

Syntax errors in your process script are indicated by an error symbol ⊗ next to the line number in which the error occurs. Syntax errors are also highlighted with a pink background in the process editor.

You can hover over either the error symbol or the highlighted syntax error to view a description of the error.



```
#
#
vEleCount = DIMSIZE(vDim);
```

⊗ The identifer is not a TM1 function at position [41, 12].

In this example, the DIMSIZ function is incorrectly spelled DIMSIZE. The incorrectly spelled function is highlighted and the error description indicates that the highlighted item is not a valid TM1 function.

Line wrapping

To enable or disable line wrapping in the Script tab on the process editor in the editor, click ⋮, then click **Enable line wrapping** or **Disable line wrapping**.

Text customization

To change the font and size of the text displayed in the Script tab on the process editor, click ⋮, then click **Change Font**.

Run a process

You can run TurboIntegrator processes directly from the content tree or from within the process editor. Regardless of where you run the process, you cannot run a process that is being edited and has any unsaved changes. You must save (or discard) changes before you run the process.

Procedure

1. To run a process from the content tree, expand the **Processes** group under the database that contains the process.
   a) Right-click the process.
   b) Click **Run Process**.
2. To run the current process in the process editor, click **Run**.
   a) You can click **Cancel** to cancel a running process in the process editor.

View process error logs

If a TurboIntegrator process execution results in errors, you can view the error logs directly in Planning Analytics Workspace.

When a process results in errors, a new window opens indicating if the process completed with errors or aborted due to errors. The window also contains the error log generated by the process.

You can resize the window to view complete lines in the log.

If a process uses the ExecuteProcess or RunProcess function to execute one or more subprocesses, and there are errors in multiple processes, the window shows each process name with an arrow that you can click to expand or contract the error log for the individual process.

Review the chores associated with a process

You can review a list of all the chores that include a specific process. From the list, you can delete the process from selected chores or edit the chores that include the process. You must be a modeler to review the chores associated with a process.

Procedure

1. In the Data tree, expand the database that contains the process.
2. Expand the **Processes** group.
3. Right-click the process, then click **Display chores involved**.

   The **Modify chores** window displays a list of all the chores that include the process.
4. Select the chores that you want to modify.
   a) Click **Chores including process_name** to select all the chores that include the process.
   b) If you don't want to modify all chores, select individual chores in the list.
5. Choose the action that you want to perform on the selected chores.
   - **Edit selected chores** - Each selected chore opens in a separate chore editor in the workspace. You can then edit each chore individually. You can open and edit up to six chores in this manner. If you select more than six chores in the list of available chores, the **Edit selected chores** option is disabled.
   - **Remove process from selected chores** - Deletes the process from each of the selected chores. There is no limit on the number of chores that you can select for this option.
6. Click **OK**.

   You can also review the chores that are associated with a specific control process.
   a) Expand the **Control objects group**.
   b) Expand the **Processes** group.
   c) Right-click the control process, then click **Display chores involved**.

Delete a process

You can delete a process directly from the content tree.

About this task

You cannot delete a process that is included in any chores. You must remove the process from all chores before you can successfully delete the process.

Procedure

1. Right-click the process in the Data tree.
2. Click **Delete process**.
3. Click **OK** to confirm the deletion.

Zero suppression in a TurboIntegrator process affects member ordering in a view

When you turn on zero suppression in a TurboIntegrator process, data in a view is ordered by when members (elements) were added to the dimension, instead of by the default dimension order (alphabetical) or by the defined subset order.

When a TurboIntegrator process is created with the parameter `pSuppressZero = 0` (suppress zero is turned off), it shows data for the dimension in descending alphabetical order or by the defined subset order, as expected. When a TurboIntegrator process is created with the parameter `pSuppressZero = 1` (suppress zero is turned on), it shows data for the dimension in the order that the members were added to the dimension.

This is an as-designed limitation based on the different types of cube data iteration associated with the suppress zeroes property on the view.

When zero suppression is off, TM1 performs an "expanse iteration" of the base cube data that visits all possible locations implied by the metadata. This type of iteration can be prohibitively slow.

When zero suppression is on, TM1 instead performs a "population iteration" that is driven by where actual data resides. This type of iteration can be much faster. However, the expanse iteration works by using member sorting indexes, while the population iteration uses member data coordinate indexes (of necessity, since that is how the data is stored.) When a new member is added, the sorting indexes of the members are reassigned to implement alphabetical ordering based on the member name. But the new member must also be assigned a new data coordinate index. This index will be above the existing range of data indexes that are already established at that time. This is the main reason for the difference in behavior.

This problem can be resolved by performing a TM1 database restart. After the database restart, the dimension will be freshly loaded into memory, at which time the sorting indexes and data coordinate indexes will match.

TurboIntegrator chores

You can create a chore to execute one or more TurboIntegrator processes in a specified sequence and at defined intervals.

A chore is composed of:

- A list of processes to be executed
- Optionally, runtime parameter values to be passed into processes upon execution from within the chore
- A start date and time for the initial execution of the chore
- An interval at which the chore is subsequently executed

Even when a chore has an execution interval defined, it can be manually executed at any time.

Once defined, chores can be activated and deactivated as required.

Access to chores functionality is controlled by your Planning Analytics Workspace role and user group security privileges. You must be a Planning Analytics Administrator or Modeler to be able to see chores in the content tree and have access to the chores shortcut menu options. To create or edit chores, you must also be a member of the Admin or DataAdmin group on the TM1 database.

Create and edit chores

You can create chores to automate the execution of TurboIntegrator processes at defined intervals.

About this task

You must be a Planning Analytics Administrator or Modeler to be able to see chores in the content tree and have access to the chores shortcut menu options. To create or edit chores, you must also be a member of the Admin or DataAdmin group on the TM1 database.

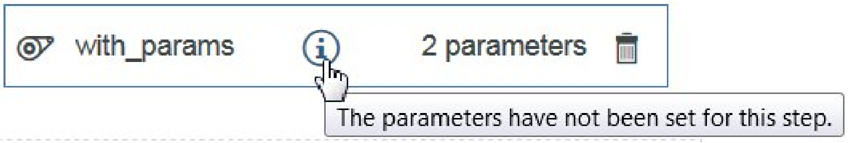
Note: To use chores, you must be running a TM1 database version 11.3 or later. TM1 database 11.3 is the version that is included with Planning Analytics version 2.0.5. If you are using a TM1 database version prior to 11.3, chores will not be available in the content tree and you cannot create or edit chores in Planning Analytics Workspace.

Procedure

1. To create a new chore:
   a) In edit mode, expand the database in the Data tree on which you want to create the new process.
   b) Right-click the **Chores** group.
   c) Click **Create chore**.
   d) You can change the database where the chore is stored by selecting a database from the list. The current database is selected by default.
   e) Enter a name for the chore.
   f) Click **Create**.
2. To edit an existing chore:
   a) Right-click the chore in the Data tree.
   b) Click **Edit chore**.
3. Specify which processes are included in the chore.
   a) Click **Add process**.
   b) Select the processes you want from the list of processes available on your TM1 database.
   c) Click **Add**.

      Processes are inserted into the chore in alphabetical order, regardless of the order in which you selected the processes.
   d) To delete a process from the chore, click the **Delete** icon 🗑for the process.
4. Set the order in which processes are executed.
   a) Click a process in the list of chore processes.
   b) Click ↑ or ↓ to move the process to the desired order in the chore. Repeat for any other processes until the desired execution order is set.
5. You must set parameter values for any processes in the chore that include parameters. The values you set are passed into the process upon execution within the chore.

   If any process includes parameters that have not been set, an Info icon warns you of this condition.



The parameters have not been set for this step.

   a) Click the parameter listing for the process (1 parameter, 2 parameters, 3 parameters, ...).
   b) Enter a value for each parameter.
   c) Click **Update**.
6. Set the **Commit each process individually** option.

   When this option is enabled, each process in the chore is committed individually, as a series of separate transactions. Any locks that are required for a given process are held only for the duration of the process in which the lock is acquired.

   When this option is disabled, all processes in the chore are processed together and committed as a single transaction. Any locks acquired by any processes in the chore are held until the last process is complete. This means locks can potentially be held for very long periods of time.
7. Specify the initial chore execution date and time, and set a subsequent execution interval.
   a) Click the **Date** box and pick the initial execution data.

   b) Click the **Time** box and set the initial execution time, using your local time.
   c) Click the **Frequency** box and pick the interval at which the chore is executed: **Daily** or **Advanced**. If you want to execute at any interval other than daily, click **Advanced**, then set the interval and click **Set Frequency**.

8. Enable the **Scheduled** option to activate the chore schedule.

   If you do not enable this option, the chore is saved on your database, but is not executed until you enable the schedule or execute the chore on demand.

9. Click **Save**.

   You can optionally click **Run** to execute the chore immediately.

Execute a chore on demand

You can execute a chore on demand at any time, regardless of the chore schedule.

Procedure

1. Right-click the chore in the Data tree.
2. Click **Run chore**.

   A status message in the Data tree region lets you know if the chore runs successfully or if errors prevent the chore from running.

Enable or disable a chore schedule

You can enable or disable the schedule in the chore editor when creating or editing a chore. You can also enable or disable a chore schedule in the content tree.

Procedure

1. Right-click the chore in the content tree.
2. Click the appropriate enable/disable schedule option.

   The options available vary depending on the current state of the chore.

   - If the chore schedule is currently enabled, the **Disable chore schedule** option is available.
   - If the chore schedule is currently disabled, the **Enable chore schedule** option is available.
   - If the chore has been created and saved, but is not yet complete (does not include any processes or does not have an execution frequency set), you won't see an enable/disable schedule option. You must complete the chore definition before you can enable or disable the schedule.

Delete a chore

You can delete a chore directly from the Data tree.

Procedure

1. Right-click the chore in the Data tree.
2. Click **Delete chore**.
3. Click **OK** to confirm the deletion.

Translate a model

You can translate a model in Planning Analytics Workspace by creating translated values for attributes. You can translate dimensions, members, cubes, attributes, views, processes, and public sets.

You create translated values by adding a caption attribute, and then by assigning caption values for every language that you want to make available. You add caption attributes and caption values by creating TurboIntegrator processes.

Planning Analytics Workspace uses the browser language setting to determine which language to display, so a user can view cubes, views, dimensions, and so on, in their own language without needing any additional configuration.

Caption attribute

A caption is an attribute that has a name of `Caption`. A caption can be either an alias type or a string type, but in Planning Analytics Workspace, the caption must have an alias type, and the value for the caption must be unique. The caption attribute must be created for every object that you want to have translations for, such as cube or set. You can create attributes with TurboIntegrator processes. Member attributes can be created in the dimension editor.

Language locale codes and behavior of the Caption attribute

International language codes that are defined by ISO 639-1 are used to identify major languages and IETF language tags to identify specific locales. For example, `fr` identifies French, while `fr-CA` identifies French (Canada).

You can assign Caption attribute values for major language codes, such as `fr`, as well as any associated specific locales, such as `fr-Fr` or `fr-CA`.

If a Caption attribute value does not exist for a given specific locale, the value of the associated major language code is retrieved. For example, if a Caption attribute value does not exist for `pt-BR`, the value for `pt` is retrieved.

If no values are found for a Caption attribute, the base default attribute value is returned.

You can see which ISO 639-1/IETF combinations are supported in the }Cultures control dimension.

Translate cube names

To display cube names in other languages, first write a process that creates the Caption attribute for all cubes on the database. Then, assign values for the cube names that you want to translate.

Procedure

1. Create a new process.
2. Click **Script**, and in the **Prolog**, create the Caption attribute as an alias attribute. Type `CubeAttrInsert( '', 'Caption', 'A');`

   To find out more, see CubeAttrInsert.
3. For each cube that you want to translate, add a CubeAttrPutS function for each language that you want to make available on the database.

   For example, if you want to display the Sales cube and Price cube in French and German, include the following four functions:

   ```
   CubeAttrPutS( 'Ventes', 'Sales', 'Caption', 'fr' );
   CubeAttrPutS( 'Vertrieb', 'Sales', 'Caption', 'de' );
   CubeAttrPutS( 'Prix', 'Price', 'Caption', 'fr' );
   CubeAttrPutS( 'Preis', 'Price', 'Caption', 'de' );
   ```

   This process creates a control cube called }LocalizedCubeAttributes with the following dimensions: }Cubes, }Cultures, }CubeAttributes.
4. Validate, save, and run the process.

What to do next

To test this process, save the book, then set the browser language to the language that you want to display. Refresh the browser and check that the correct cube name is displayed in the content tree.

Translate dimension and hierarchy names

You can display dimension and hierarchy names in other languages by using processes. Create a process that creates the Caption attribute for all dimensions on the database, and then assign values for the dimension and hierarchy names that you want to translate.

Procedure

1. Create a new process.
2. Click **Script**, and in the **Prolog**, create the Caption attribute as an alias attribute.

   Type

   ```
   DimensionAttrInsert( '', 'Caption', 'A');
   ```

   To find out more, see DimensionAttrInsert.

   This function creates a control cube called }DimensionAttributes with the following dimensions: }Dimensions, }DimensionAttributes.
3. For each dimension and hierarchy name that you want to translate, add a DimensionAttrPutS function for each language that you want to make available.

   For example, if you want to display the name of the Model dimension in French and Portuguese, include the following functions:

   ```
   DimensionAttrPutS( 'Modèle', 'Model', 'Caption', 'fr' );
   DimensionAttrPutS( 'Modelo', 'Model', 'Caption', 'pt' );
   ```

   If you want to display a hierarchy named Engine Size in the Model dimension in French, include the following function:

   ```
   DimensionAttrPutS( 'Taille du moteur', 'Model:Engine Size', 'Caption', 'fr' );
   ```

   DimensionAttrPutS creates a control cube called }LocalizedDimensionAttributes with the following dimensions: }Dimensions, }Cultures, and }DimensionAttributes.
4. Validate, save, and run the process.

What to do next

To test this process, save the book, then set the browser language to the language that you want to display. Refresh the browser and check that the correct dimension name is displayed in the content tree, in the dimension editor, and in a view.

Translate members

To display dimension member names in other languages, create a process that creates the Caption attribute for all member names on the database. Then, assign values for the member names that you want to translate.

Procedure

1. Create a new process.
2. Click **Script**, and in the **Prolog**, create the Caption attribute as an alias attribute. Enter the following function:

   ```
   AttrInsert( '<dim_name>', '', 'Caption', 'A');
   ```

   This function creates the Caption attribute as an alias attribute for the members of the *<dim_name›*on the database. To find out more, see AttrInsert.
3. For each member name that you want to translate, add an AttrPutS function for each language that you want to make available on your database.

For example, if you want to display the January member in French, German, and Portuguese, include the following functions:

```
AttrPutS('Janvier', 'Month', 'January', 'Caption', 'fr');
AttrPutS('Januar', 'Month', 'January', 'Caption', 'de');
AttrPutS('Janeiro', 'Month', 'January', 'Caption', 'pt');
```

The first time that you add an attribute with the fourth parameter for a Culture, a cube is created: }LocalizedElementAttributes_,*<dim_name>* with the following dimensions: *<dim_name>*, }Cultures, }ElementAttributes_*<dim_name>*

Tip: You can also use the ElementAttrPutS function to add attributes to members. This attribute produces the same results as `AttrPutS`

4. To translate a member that exists only in a hierarchy other than the hierarchy with the same name as the dimension, see the following example:

```
AttrPutS( 'Rouge', 'model:Color', 'Red', 'Caption', 'fr' );
```

This example displays the caption Rouge when the member Red, in the Color hierarchy, which is part of the model dimension, is shown, and the browser is set to French.

5. Validate, save, and run the process.

What to do next

To test this process, save the book, then set the browser language to the language that you want to display. Refresh the browser and check that the members are displayed in the correct language in the content tree and in the dimension editor.

Translate set names

To display set names in other languages, create a process that creates the Caption attribute for all member names in the dimension, and then assigns values for the set names that you want to translate. This option is available only if your organisation has deployed IBM Planning Analytics version 2.0.4 or later.

Procedure

1. Create a new process.
2. In the Prolog, create the Caption attribute as an alias attribute.

   Type

```
SubsetAttrInsert('<dim_name>', '', 'Caption', 'A');
```

To find out more, see SubsetAttrInsert.

This function creates a control cube called SubsetAttributes_*<dim_name>* with the following dimensions: }Subsets_*<dim_name>*, }SubsetAttributes_*<dim_name>*.

3. For each subset that you want to translate, add a SubsetAttrPutS function for each language that you want to make available on your database:

```
SubsetAttrPutS( '<String>', '<dim_name>', '<set_name>', 'Caption', '<Lang_Locale_Code>' );
```

For example, if you want to display the Sales set in the Departments dimension in French, include the following function:

```
SubsetAttrPutS( 'Ventes', 'Departments', 'Sales', 'Caption', 'fr' );
```

This creates a control cube called LocalizedSubsetAttributes_*<dim_name>* with the following dimensions: }Subsets_*<dim_name>*, }Cultures, }SubsetAttributes_*<dim_name>*.

What to do next

To test this process, save the book, then set the browser language setting to the language that you want to display. Refresh the browser and add a view that contains the translated set. Check that the set name is displayed in the correct language.

Translate view names

To display view names in other languages, create a process that creates the Caption attribute for all views in the cube. Then, assign values for the view names that you want to translate.
This option is available only if your organization deployed IBM Planning Analytics version 2.0.4 or later.

Note: View names can be translated only for views that are saved in the database. Views that are created in IBM Planning Analytics Workspace are not saved in the database. They are saved in the book.

Procedure

1. Create a new process.
2. Click **Script**, and in the **Prolog**, create the Caption attribute as an alias attribute.

   Type

   ```
   ViewAttrInsert('<cube_name>', '', 'Caption', 'A');
   ```

   This function creates a control cube that is called ViewAttributes_*<cube_name>* with the following dimensions: }Views_*<cube_name>*, }ViewAttributes_*<cube_name>*. To find out more, see ViewAttrInsert.
3. For each view that you want to translate, add a ViewAttrPutS function for each language that you want to make available on your database:

   ```
   ViewAttrPutS( '<String>', '<cube_name>', '<view_name>', 'Caption', '<Lang_Locale_Code>');
   ```

   For example, to translate a view called Price into French on a cube that is called SalesCube, include the following function:

   ```
   ViewAttrPutS( 'Prix', 'SalesCube', 'Price', 'Caption', 'fr' );
   ```

   This function creates a control cube that is called LocalizedViewAttributes_*<cube_name>* with the following dimensions: }Views_*<cube_name>*, }Cultures, }ViewAttributes_*<cube_name>*.

What to do next

To test this process, save the book, then set the browser language to the language that you want to display. Refresh the browser and find the view with the translated name in the content tree. Check that the view name is displayed in the correct language.

Translate process names

To display process names in other languages, create a process that creates the Caption attribute for all processes on the database, and then assigns values for the process names that you want to translate.This option is available only if your organization has deployed IBM Planning Analytics version 2.0.4 or later.

Procedure

1. Create a new process.
2. Click **Script**, and in the **Prolog**, create the Caption attribute as an alias attribute. Enter the following function:

   ```
   ProcessAttrInsert( '', 'Caption', 'A');
   ```

   This function creates a control cube called }ProcessAttributes with the following dimensions: }Processes, }ProcessAttributes. To find out more, see ProcessAttrInsert.

3. For each process that you want to translate, add a ProcessAttrPutS function for each language that you want to make available on the database.

   For example, the following process displays the name of Import data process in French as *Importer des données*.

   ```
   ProcessAttrPutS('Importer des données', 'Import data', 'Caption', 'fr');
   ```

4. Validate, save, and run the process.

What to do next

To test this process, save the book, then set the browser language setting to the language that you want to display. Refresh the browser and check that the correct process name is displayed in the content tree.

Chapter 7. Administer IBM Planning Analytics Workspace

Administering IBM Planning Analytics Workspace includes managing users, administering databases, migrating assets, and other administrative actions. You access the **Administration** page from the Administration tile on the Planning Analytics Workspace **Home** page.

Administer users and groups

To administer users in IBM Planning Analytics Workspace, you must be logged in to Planning Analytics Workspace as an **administrator**. If you have the administrator role and meet all of the requirements, you can invite people who are in the same organization (account) as you to become users in Planning Analytics Workspace on cloud or upload users to Planning Analytics Workspace Local.

Administer users on cloud

To administer users in IBM Planning Analytics Workspace on cloud, you must understand the intersection between subscriptions, environments, and roles. Your user profile must also have a few more characteristics.

What is the profile of an administrator who can administer users on cloud?

The user profile of an administrator on cloud shows the full name, user ID, contact email, role, subscription, environments, and groups. As an administrator of users on cloud, you must also have the following characteristics:

- You must be a member of the organization for which you want to administer users. To see the organizations that you are a member of, see "Manage user invitation capability (cloud only)" on page 268.
- You must be a subscription administrator. To check, look for **Subscription administrator** beside the **Subscription** field in your user profile.
- You are a subscription administrator only for the first organization that you were added to. You cannot be a subscription administrator for multiple organizations.

What's an organization versus an environment?

An organization has one primary environment and in addition, can have other environments. For example, your primary environment is your production environment and you can also have a non-production, or test, environment. A user can have access to multiple environments. A user can have one role and one subscription in an environment.

Important: When a user who is a member of more than one environment accepts an email invitation to a newly created environment, the provisioning of the user in the new environment might not be complete before the user accepts the invitation. Waiting a short time and accepting the invitation again redirects the user to the new environment or to an environment selector with the new environment in the list of available environments.

What do I need to know about subscriptions?

In Planning Analytics Workspace on cloud, the subscription determines the types of roles that a user can have.

In the following table, a user with the subscription shown in the column heading can have the roles in the row marked with a check mark.

For example, if a user has an IBM Planning Analytics User subscription, then they can be assigned the Consumer or Analyst role.

Table 8. The intersection between subscriptions and roles

	Planning Analytics Modeler subscription	**Planning Analytics User subscription**	**Planning Analytics Explorer subscription**
Consumer role	✓	✓	✓
Analyst role	✓	✓	
Modeler role	✓		
Administrator role	✓		

Workflow

Follow these steps to manage a group of new users on cloud:

1. **Invite user** to join Planning Analytics Workspace. The user has a status of *Invited but not activated*. The user receives an invitation by email. If a user accepts the invitation, the user has a status of *Active*. For more information, see "Add and invite a user (cloud only)" on page 270.
2. **Upload users** using a CSV file and then invite them to use Planning Analytics Workspace. In Planning Analytics Workspace on cloud, all users have the Analyst role and the same subscription as you. For more information, see "Upload a file to add or remove multiple users (cloud only)" on page 274.
3. **Create** groups of users in the **Groups** tab. For more information, see "Administer groups" on page 287.
4. Optional: **Upload groups** in the **Groups** tab by dropping a CSV file on to the dialog box or tapping the dialog box to browse for a local CSV file. For more information, see "Administer groups" on page 287.
5. **Export users** to a CSV file. For more information, see "Export users" on page 288.
6. Click **Export groups** in the **Groups** tab to save a CSV file of existing groups to your local file system. This is a great way to create a CSV file to add users to existing groups.
7. Optional: **Manage roles** for several users at one time. This is useful if you upload a group of users but want to assign them different roles. For more information, see "Change a user's role" on page 289.
8. Optional: **Manage subscriptions** for several users at one time. This approach is useful if you upload a large group of users but want to change individual subscriptions. For more information, see "Change a user's subscription (cloud only)" on page 280.
9. Optional: **Manage environments** for a user. For more information, see "Change a user's environment" on page 291.
10. Optional: **Manage user accounts** to add or delete a user as a subscription administrator. For more information, see "Manage user invitation capability (cloud only)" on page 268.

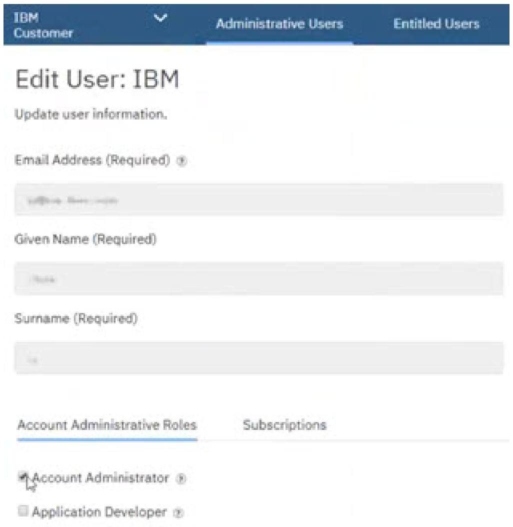
Manage user invitation capability (cloud only)

Your organization (account) can have multiple users that are account administrators. Account administrators can invite other users to your organization.

Manage user invitation capability in Planning Analytics Workspace

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

[redacted]

Given Name (Required)

[redacted]

Surname (Required)

[redacted]

☑ Account Administrator

□ Application Developer

About this task

A user can be an account administrator only for the first organization that they were added to.

You can see the organization that the user can be an account administrator for on the **Administrative Users** page or the **Entitled Users** page.

You can also perform this task to see the organization that a user belongs to.

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users & Groups**.
3. Click **IBM Subscriptions Management**.

   The organization (account) that you belong to is displayed in the upper left of the window.
4. Click **Entitled Users**.

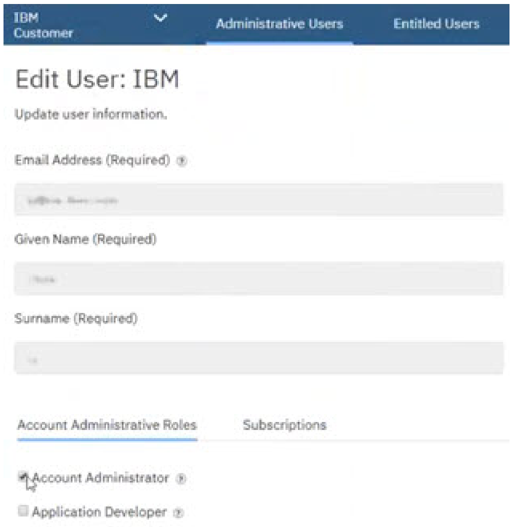
   The list that displays shows all the users that are members of your organization. For each user, their name, email address, organization, subscription, and entitlement status is displayed.
5. In the list, click the name of the user that you want to make an account administrator.
6. In the **Edit User** window, click **Account Administrative Roles**, and select the **Account Administrator** check box.

   Remember: The user can be an account administrator only for the first organization that they were added to.
7. Click **Save and Close**.

Manage user invitation capability in Planning Analytics Workspace Classic

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

☑ Account Administrator

□ Application Developer

About this task

A user can be an account administrator only for the first organization that they were added to.

You can see the organization that the user can be an account administrator for on the **Administrative Users** page or the **Entitled Users** page.

You can also perform this task to see the organization that a user belongs to.

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. On the **Users** page, click **Manage user accounts**.

   The organization (account) that you belong to is displayed in the upper left of the window.
3. Click **Entitled Users**.

   The list that displays shows all the users that are members of your organization. For each user, their name, email address, organization, subscription, and entitlement status is displayed.
4. In the list, click the name of the user that you want to make an account administrator.
5. In the **Edit User** window, click **Account Administrative Roles**, and select the **Account Administrator** check box.

   Remember: The user can be an account administrator only for the first organization that they were added to.
6. Click **Save and Close**.

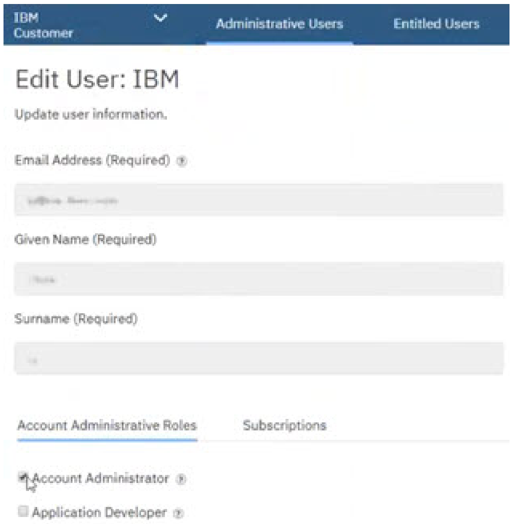
Add and invite a user (cloud only)

To add and invite users one at a time, follow these instructions.

Add and invite a user in Planning Analytics Workspace

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

☑ Account Administrator

□ Application Developer

About this task

If you invite a single user at a time, you must invite the user to your primary environment first. After the user accepts the invitation to the primary environment, you can add them to other environments. A user becomes active after they accept the invitation.

Procedure

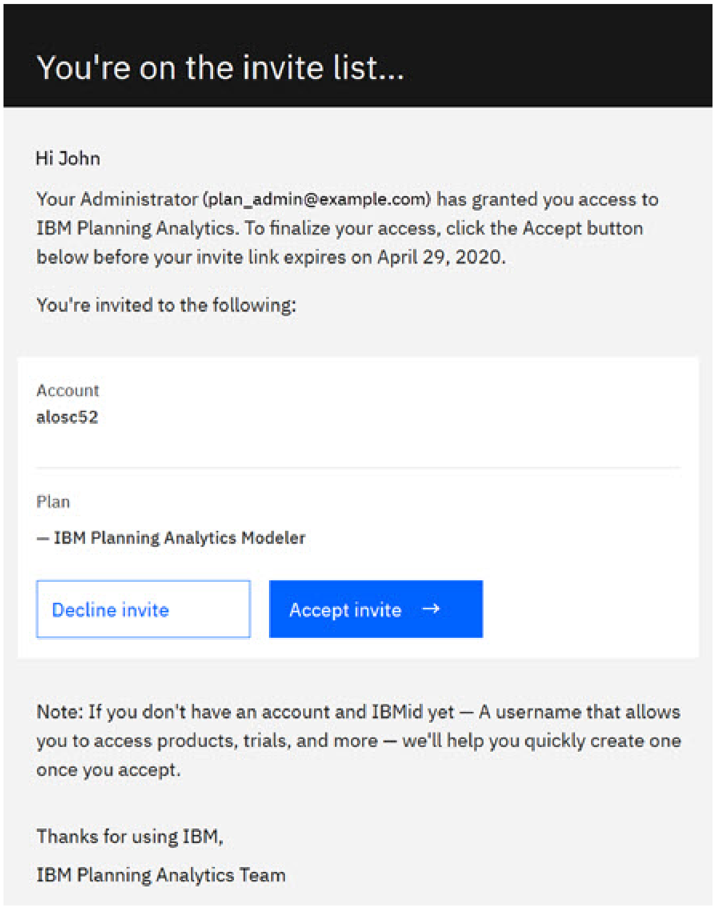
1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users & Groups**.
3. Click the **Users** tab.
4. Click **Invite a new user** + and enter their first name, last name, and email address.
5. Select a role.

   The Analyst role is selected by default. You can change the role later. For more information about roles, see "User roles" on page 31.
6. Select a subscription for the user, and then click **Submit**.

   After a user is invited, their status is **Invited but not activated**.

Results

When you send an invitation, the user receives an email similar to this. The invitation clearly shows the email address of the administrator who sent the invitation, as well as the account and plan associated with the invitation.



You're on the invite list...

Hi John

Your Administrator (plan_admin@example.com) has granted you access to IBM Planning Analytics. To finalize your access, click the Accept button below before your invite link expires on April 29, 2020.

You're invited to the following:

Account
alosc52

Plan
— IBM Planning Analytics Modeler

Note: If you don't have an account and IBMid yet — A username that allows you to access products, trials, and more — we'll help you quickly create one once you accept.

Thanks for using IBM,

IBM Planning Analytics Team

The invitation is valid for 28 days. A user cannot access Planning Analytics Workspace if they do not accept the invitation. If the user declines the invitation or does not accept within 28 days, the invitation is revoked and you must send a new invitation to grant access to the user.

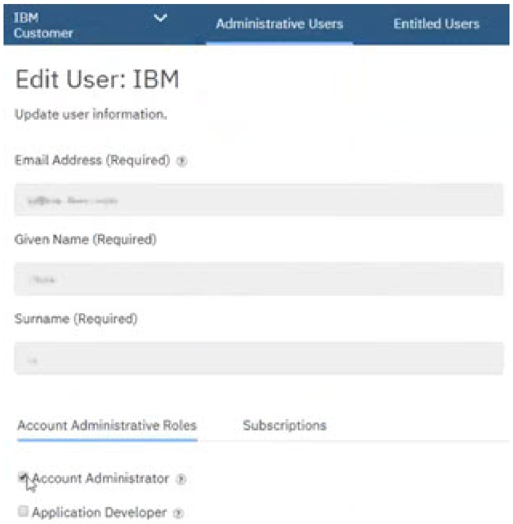
Note that invitation and confirmation emails come from several different IBM systems and email accounts. Tell your users to expect these emails.

After the user accepts the invitation and logs in to Planning Analytics Workspace, their status changes to **Active**.

Add and invite a user in Planning Analytics Workspace Classic

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



IBM Customer

Administrative Users | Entitled Users

Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

Account Administrative Roles | Subscriptions

☑ Account Administrator

□ Application Developer

About this task

If you invite a single user at a time, you must invite the user to your primary environment first. After the user accepts the invitation to the primary environment, you can add them to other environments. A user becomes active after they accept the invitation.

Procedure

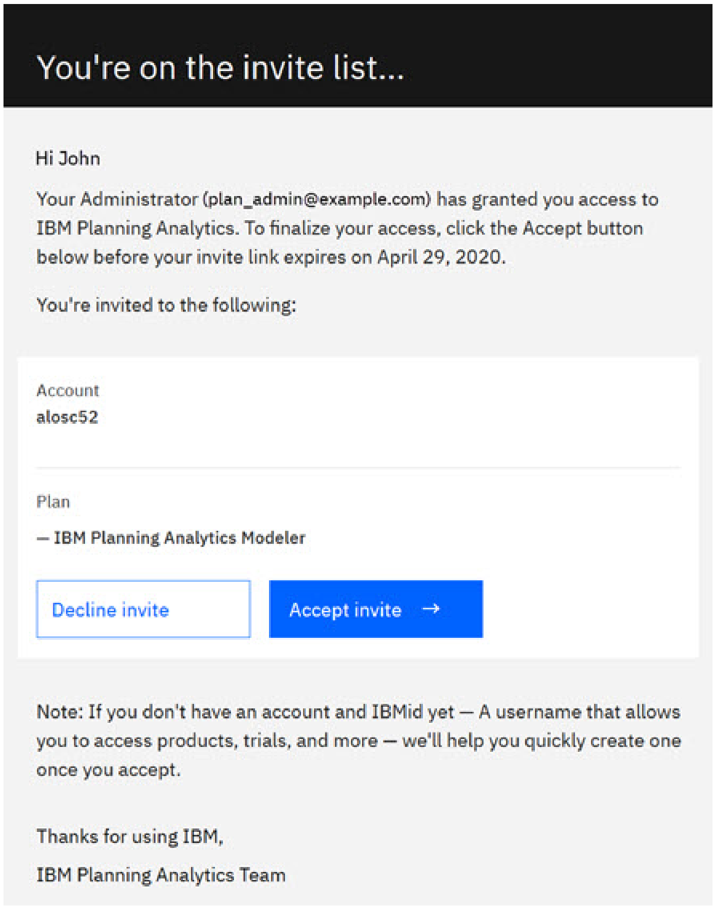
1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the **Users** tab.
3. Click **Invite user** and enter their first name, last name, and email address.
4. Select a role.

   The Analyst role is selected by default. You can change the role later. For more information about roles, see "User roles" on page 31.
5. Select a subscription for the user, and then click **Invite**.

   After a user is invited, their status is **Invited but not activated**.

Results

When you send an invitation, the user receives an email similar to this. The invitation clearly shows the email address of the administrator who sent the invitation, as well as the account and plan associated with the invitation.



You're on the invite list...

Hi John

Your Administrator (plan_admin@example.com) has granted you access to IBM Planning Analytics. To finalize your access, click the Accept button below before your invite link expires on April 29, 2020.

You're invited to the following:

Account
alosc52

Plan
— IBM Planning Analytics Modeler

Note: If you don't have an account and IBMid yet — A username that allows you to access products, trials, and more — we'll help you quickly create one once you accept.

Thanks for using IBM,
IBM Planning Analytics Team

The invitation is valid for 28 days. A user cannot access Planning Analytics Workspace if they do not accept the invitation. If the user declines the invitation or does not accept within 28 days, the invitation is revoked and you must send a new invitation to grant access to the user.

Note that invitation and confirmation emails come from several different IBM systems and email accounts. Tell your users to expect these emails.

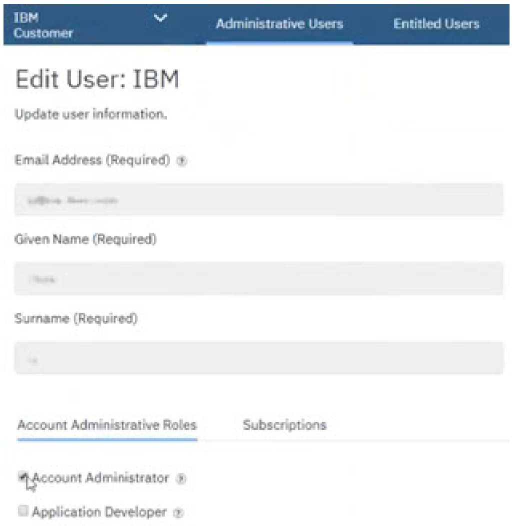
After the user accepts the invitation and logs in to Planning Analytics Workspace, their status changes to **Active**.

Upload a file to add or remove multiple users (cloud only)

To add or remove multiple users, upload a file that includes user login IDs and ADD or REMOVE directives. Users that are added receive an invitation to the environment specified in the file.

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

☑ Account Administrator

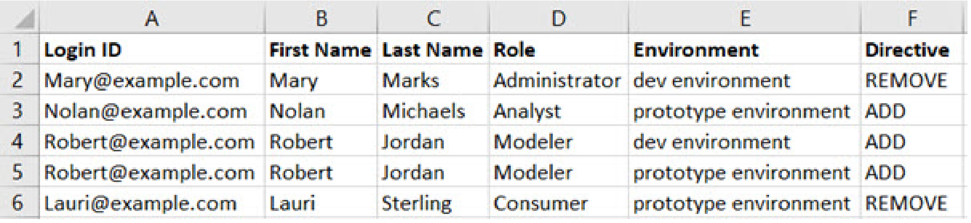
□ Application Developer

About this task

- To add and/or remove multiple users in Planning Analytics Workspace on cloud, create a CSV file that contains one line for each user with the following information: Login ID, first name, last name, role, environment, directive.

  The login ID is the email address that is used to sign in to Planning Analytics Workspace. Login ID, first name, and last name are mandatory. All other attributes are optional. If you omit role, it defaults to Analyst. If you omit environment, it defaults to the environment that you are logged in to. If you omit directive, it defaults to ADD.

  Note: When uploading users from a .csv file, please save the file with the CSV UFT-8 format encoding to ensure that special characters are imported.

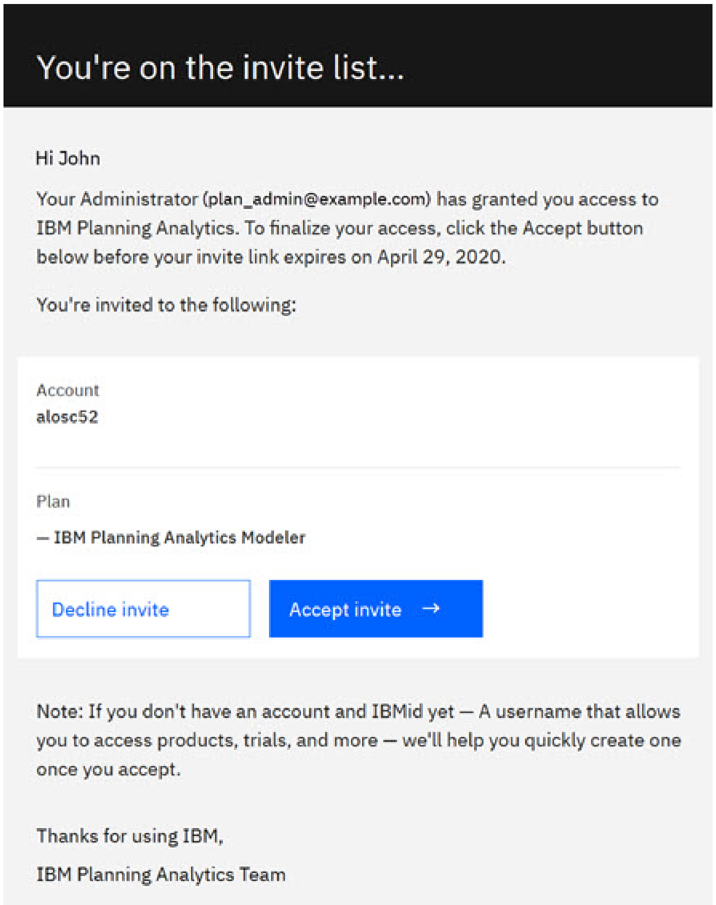


	A	B	C	D	E	F
1	**Login ID**	**First Name**	**Last Name**	**Role**	**Environment**	**Directive**
2	Mary@example.com	Mary	Marks	Administrator	dev environment	REMOVE
3	Nolan@example.com	Nolan	Michaels	Analyst	prototype environment	ADD
4	Robert@example.com	Robert	Jordan	Modeler	dev environment	ADD
5	Robert@example.com	Robert	Jordan	Modeler	prototype environment	ADD
6	Lauri@example.com	Lauri	Sterling	Consumer	prototype environment	REMOVE

- If you invite multiple users at a time, the number of users that you can invite is limited to the number of actual available Planning Analytics Workspace subscriptions you purchased, exclusive of any overage allowances. For example, if you have seven subscriptions available with unlimited overage, you can invite up to seven users. If you invite a number of users greater than the number of actual available subscriptions, the invitation fails.
- You must invite users to your primary environment first and they must be active in the primary environment before you can upload them to any secondary environments.
- You cannot assign a user to an environment that they already belong to, nor to an environment that does not exist.
- You can add users to multiple secondary environments by using a single CSV file.
- A user can have different roles in different environments. For more information about roles, see "User roles" on page 31.
- You can use file upload to remove users only from a non-primary environment. To remove a user from the primary environment, see "Remove a user (cloud only)" on page 277. You must remove a user from all non-primary environments before you remove them from the primary environment.
- You can use a file to add or remove up to 100 users at a time.

Results

When you send an invitation, the user receives an email similar to this. The invitation clearly shows the email address of the administrator who sent the invitation, as well as the account and plan associated with the invitation.



You're on the invite list...

Hi John

Your Administrator (plan_admin@example.com) has granted you access to IBM Planning Analytics. To finalize your access, click the Accept button below before your invite link expires on April 29, 2020.

You're invited to the following:

Account
alosc52

Plan
— IBM Planning Analytics Modeler

Note: If you don't have an account and IBMid yet — A username that allows you to access products, trials, and more — we'll help you quickly create one once you accept.

Thanks for using IBM,

IBM Planning Analytics Team

The invitation is valid for 28 days. A user cannot access Planning Analytics Workspace if they do not accept the invitation. If the user declines the invitation or does not accept within 28 days, the invitation is revoked and you must send a new invitation to grant access to the user.

Note that invitation and confirmation emails come from several different IBM systems and email accounts. Tell your users to expect these emails.

After the user accepts the invitation and logs in to Planning Analytics Workspace, their status changes to **Active**.

Upload a file to add or remove multiple users in Planning Analytics Workspace

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users & Groups**.
3. Click the **Users** tab.
4. Click **Upload users** ⤒.
5. Drag the CSV file to the **Drop your .csv file here** region 🗎, or browse to the file location and import the file.

   The file is uploaded and you see a success message.
6. Click **OK**.

The users that you added with the ADD directive in the CSV file appear on the **Users** tab with a status of **Not invited yet**.

7. To invite users that you uploaded with the ADD directive, select the users with a status of **Not invited yet**.
   a) Click **Send invitation** in the **Applicable actions** list.
   b) Select a subscription for the invited users.
   c) Click **Submit**.
8. To delete users that you uploaded with the REMOVE directive, select the users.
   a) Click **Delete users** in the **Applicable actions** list.
   b) Click **OK**.

Upload a file to add or remove multiple users in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the **Users** tab.
3. Click **Upload users**.
4. Drag the CSV file to the **.csv** icon, or browse to the file location and import the file.

   The file is uploaded and you see a success message.
5. Click **OK**.

   The users that you added with the ADD directive in the CSV file appear on the **Users** tab with a status of **Not invited yet**.
6. To invite users that you uploaded with the ADD directive, select the users with a status of **Not invited yet**. Click the circle next to a user name to select an individual user. Click the circle ○ at the top of the list to select all users. To select multiple adjacent users, click the circle next to the first user and then press Shift + click the circle next to the last user.

   A user is selected when a check mark in the circle ✓ accompanying the user name.
   a) Click **Invite uploaded users (100 users max)** in the **Applicable actions** list.
   b) Click **OK**.
7. To delete users that you uploaded with the REMOVE directive, select the users.
   a) Click **Delete users** in the **Applicable actions** list.
   b) Click **OK**.

Remove a user (cloud only)

As an administrator, you can remove users from IBM Planning Analytics Workspace on cloud.

Note: Removing a user's subscription to the Planning Analytics on Cloud service also renames the user's personal folder to **Unknown**. All the workspace assets of the removed user remain in the renamed folder. As an administrator, you can see but cannot open assets in the personal folder of a user that has been removed. You must move the assets from the removed user's personal folder and place them in your own personal folder or the Shared folder to open them.

Procedure

1. If you want to completely remove a user from all subscriptions to IBM products, see "Remove a user from all subscriptions" on page 278.
2. If the user has access to multiple subscriptions from other IBM products that they want to keep, see "Remove a user from Planning Analytics Workspace only " on page 279.

Remove a user from all subscriptions

You can remove a user from all IBM subscriptions.

Before you begin

- You must be a member of the same organization as the users that you are removing.
- You must be a subscription administrator.

Results

- As of Planning Analytics Workspace 2.0.54, the user is automatically removed from all environments within the organization. You no longer need to remove the user from secondary environments before removing the user from the primary environment.
- Removing a user’s subscription to the Planning Analytics on Cloud service also renames the user's personal folder to **Unknown**. All the workspace assets of the removed user remain in the renamed folder. As an administrator, you can see but cannot open assets in the personal folder of a user that has been removed. You must move the assets from the removed user's personal folder and place them in your own personal folder or the Shared folder to open them.
- The user will not have access to Planning Analytics Workspace and will no longer consume a Planning Analytics subscription.
- The user is no longer a member of the organization and can be invited to be a member of another organization.
- The user will no longer have access to any other IBM subscriptions that they might be entitled to with this subscription management service.

Remove a user from all subscriptions in Planning Analytics Workspace

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users $ Groups**.
3. Click **Users**.
4. Click **IBM Subscriptions Manager**.

   The organization (account) that you belong to opens in the upper left corner of the window.
5. Click **Entitled Users**.

   The list shows all the users that are members of your organization. For each user, you can see their name, email address, organization, subscription, and entitlement status.
6. Click ⌄ adjacent to the user that you want to remove.
7. Click **Remove Access**.
8. Click **OK** to confirm.

   Allow several minutes for the **Users** tab to update.

Remove a user from all subscriptions in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your username, and then click **Administer**.
2. Click the **Users** tab.
3. On the **Users** tab, click **Manage user accounts**.

   The organization (account) that you belong to opens in the upper left corner of the window.
4. Click **Entitled Users**.

The list shows all the users that are members of your organization. For each user, you can see their name, email address, organization, subscription, and entitlement status.

5. Click ⌄ adjacent to the user that you want to remove.
6. Click **Remove Access**.
7. Click **OK** to confirm.

   Allow several minutes for the **Users** tab to update.

Remove a user from Planning Analytics Workspace only

You can remove a user from only the Planning Analytics on Cloud service if a user has access to multiple IBM subscriptions.

Before you begin

- You must be a member of the same organization as the users that you are removing.
- You must be a subscription administrator.

Results

- As of Planning Analytics Workspace 2.0.54, the user is automatically removed from all environments within the organization. You no longer need to remove the user from secondary environments before removing the user from the primary environment.
- Removing a user’s subscription to the Planning Analytics on Cloud service also renames the user's personal folder to **Unknown**. All the workspace assets of the removed user remain in the renamed folder. As an administrator, you can see but cannot open assets in the personal folder of a user that has been removed. You must move the assets from the removed user's personal folder and place them in your own personal folder or the Shared folder to open them.
- The user will not have access to Planning Analytics Workspace and will no longer consume a Planning Analytics subscription.

Remove a user from Planning Analytics Workspace only in Planning Analytics Workspace

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users & Groups** tab.
3. Click the Users tab.
4. On the **Users** page, click **IBM Subscriptions Management**.

   The organization (account) that you belong to opens in the upper left corner of the window.
5. Click **Entitled Users**.

   The list shows all the users that are members of your organization. For each user, you can see their name, email address, organization, subscription, and entitlement status.
6. Click the name of the user that you want to remove.
7. In the **Edit User** window, click **Subscriptions**, and clear the **Planning Analytics Subscription** (Modeler, User, or Explorer) check box.
8. Click **Save and Close**.

   Allow several minutes for the **Users** tab to update.

Remove a user from Planning Analytics Workspace only in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your username, and then click **Administer**.
2. Click the **Users** tab.

3. On the **Users** page, click **Manage user accounts**.

   The organization (account) that you belong to opens in the upper left corner of the window.
4. Click **Entitled Users**.

   The list shows all the users that are members of your organization. For each user, you can see their name, email address, organization, subscription, and entitlement status.
5. Click the name of the user that you want to remove.
6. In the **Edit User** window, click **Subscriptions**, and clear the **Planning Analytics Subscription** (Modeler, User, or Explorer) check box.
7. Click **Save and Close**.

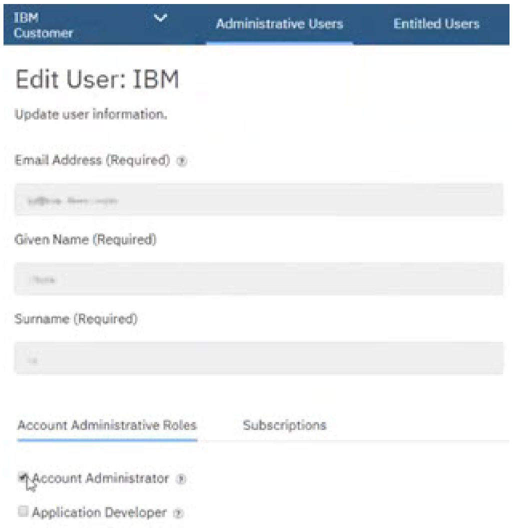
   Allow several minutes for the **Users** tab to update.

Change a user's subscription (cloud only)

You can change the subscription for one or more users at a time. Subscriptions apply only to IBM Planning Analytics Workspace on cloud.

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

☑ Account Administrator

□ Application Developer

About this task

Your IBM Planning Analytics license determines the subscriptions that are available for you to assign to a user. For example, if your organization has a standard subscription, then you can assign one of the following subscriptions to a user:

- IBM Planning Analytics Explorer
- IBM Planning Analytics User
- IBM Planning Analytics Modeler

Change a user's subscription in Planning Analytics Workspace

Procedure

1. On the **Welcome** page, the **Administration** tile.
2. Click **Users & Groups**.
3. Click **IBM Subscriptions Management**.

4. Click the user for whom you want to change subscriptions.
5. On the **Edit user** window, click the **Subscriptions** tab.
6. Select the subscriptions for the user, then click **Save and close**.

Change a user's subscription in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the **Users** tab.
3. Beside **Manage user accounts**.
4. Click the user for whom you want to change subscriptions.
5. On the **Edit user** window, click the **Subscriptions** tab.
6. Select the subscriptions for the user, then click **Save and close**.

View your subscription consumption (cloud only)

As an administrator, you can see view your total Planning Analytics subscription entitlement and determine how many subscriptions of your total entitlement are available.

View your subscription consumption in Planning Analytics Workspace

Procedure

1. On the **Welcome** page, click the **Administration** tile.
2. Click **Users & Groups**.
3. On the **IBM Subscription Management** page.

   The organization (account) that you belong to is displayed in the upper left of the window.
4. Click **Subscriptions**.

   You can see all the subscriptions and determine how many subscriptions of your total entitlement are available.

   Overage accounts indicate that your organization is using more than it is entitled to. These accounts are subject to overage fees according to your contractual agreement.

View your subscription consumption in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your username, and then click **Administer**.
2. Click the **Users** tab.
3. On the **Users** page, click **Manage user accounts**.

   The organization (account) that you belong to is displayed in the upper left of the window.
4. Click **Subscriptions**.

   You can see all the subscriptions and determine how many subscriptions of your total entitlement are available.

   Overage accounts indicate that your organization is using more than it is entitled to. These accounts are subject to overage fees according to your contractual agreement.

Understand who is a default member of the ADMIN security group in TM1

All users with the Administrator role in Planning Analytics Workspace are included in the Cognos group with `CAMID("pans:g:Subscription Administrators")`.

When a TM1 database is created, a new CAM associated group entry is created in the control cube `}ClientCAMAssociatedGroups`, which maps the Cognos group

`CAMID("pans:g:Subscription Administrators")` to the generic TM1 security group `ADMIN`. When a user is assigned the Administrator role in Planning Analytics Workspace, they are implicitly added to the `CAMID("pans:g:Subscription Administrators")` group, and are automatically added to the ADMIN security group in TM1.

All TM1 security groups, including the ADMIN group, are described in the TM1 Operations Guide.

Administer local users

To administer users in IBM Planning Analytics Workspace Local, you must have the **Administrator** role. Your user profile shows your full name, user ID, role, and groups that you belong to.

Workflow

Follow these steps to manage a group of users locally:

1. **Upload users** by dropping a CSV file on to the dialog box or tapping the dialog box to browse for a local CSV file. For more information, see "Add users (local only)" on page 282.
2. **Create** groups of users in the **Groups** tab. For more information, see "Administer groups" on page 287.
3. Optional: **Upload groups** in the **Groups** tab by dropping a CSV file on to the dialog box or tapping the dialog box to browse for a local CSV file. For more information, see "Administer groups" on page 287.
4. **Export users** to a CSV file. For more information, see "Export users" on page 288.
5. Click **Export groups** in the **Groups** tab to save a CSV file of existing groups to your local file system. This is a great way to create a CSV file to add users to existing groups.
6. Optional: **Manage roles** for several users at one time. This is useful if you upload a group of users but want to assign them different roles. For more information, see "Change a user's role" on page 289.
7. Optional: **Activate or deactivate a user**. For more information, see "Activate or deactivate a user (local only) " on page 285.
8. Optional: **Delete users**. For more information, see "Delete a user (local only)" on page 285.
9. Optional: **Delete multiple users** by using a CSV file. For more information, see "Delete multiple users (local only)" on page 286.

Add users (local only)

To add multiple users to IBM Planning Analytics Workspace Local, you upload a list of users and then can you activate, deactivate, or delete them.

Before you begin

To administer users in IBM Planning Analytics Workspace Local, you must have the **Administrator** role.

About this task

Settings for the ENABLE_USER_IMPORT configuration parameter affect when a user can log in to Planning Analytics Workspace Local.

For more information, see Planning Analytics Workspace configuration parameters (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_config_file.html).

What to do next

- "Administer groups" on page 287.
- "Change a user's role" on page 289.
- "Activate or deactivate a user (local only) " on page 285.
- "Delete a user (local only)" on page 285.

Add users in Planning Analytics Workspace

Procedure

1. Create a CSV file that contains the users that you want to upload.

   The CSV file can have a header row. If you use a header row, you can order the user information however you want as long as the names of the columns match the supported names. For example:

   ```
   Login ID, First Name, Last Name, Email
   Planning Analytics/Mary Smith,Mary,Smith,msmith@ca.ibm.com
   Planning Analytics/Robert Smith,Robert,Smith,roberts@ca.ibm.com
   Planning Analytics/Kevin Alexander,Kevin,Alexander,kalex@ca.ibm.com
   ```

   The CSV file must contain one line for each user with the following information: Login ID, First Name, Last Name, Role, Email, Status. Login ID, First Name, and Last Name are required. Role, Email, and Status are optional.

   If you omit Role, it defaults to Analyst. For information about roles, see "User roles" on page 31.

   If you omit Email, it defaults to Login ID. If the Login ID is not a valid email address, the user will be added with a state of `Not invited yet` and cannot log in to Planning Analytics Workspace Local.

   Status can be `Active`, `Inactive`, or `Inactive_suspended`. If you omit status, it defaults to Suspended. A user that has a status of Suspended can't log in to Planning Analytics Workspace until you activate them.

   Note: Text values in the CSV file such as role and status must be in English.

   If the CSV file does not have a header row, use commas to ensure that the information is in the correct layout. For example:

   ```
   Planning Analytics/Mary Smith,Mary,Smith,,msmith@ca.ibm.com,Active
   Planning Analytics/Robert Smith,Robert,Smith,,roberts@ca.ibm.com,Suspended
   Planning Analytics/Kevin Alexander,Kevin,Alexander,Administrator,kalex@ca.ibm.com,Active
   ```

2. On the **Welcome** page, click the **Administration** tile.

   Note: If you don't see the **Administer** option, then you are not logged in as administrator and you cannot add users in Planning Analytics Workspace Local.

3. Click the **Users & Groups** tab.
4. Click the **Users** tab.
5. Click **Upload users from a .csv file** ⤒.
6. Drag the CSV file onto the **Drop your .csv file here** region 🗎, or browse to the file location and import the file.

   The file is uploaded and you see a success message.

7. Click **OK**.

   The users appear under the **Users** tab with the status that you specified in your CSV file.

   Note: Status appears as Active, Suspended, Invited, or Not invited yet depending on whether you supplied a Status in your CSV file.

What to do next

Add users in Planning Analytics Workspace Classic

Procedure

1. Create a CSV file that contains the users that you want to upload.

The CSV file can have a header row. If you use a header row, you can order the user information however you want as long as the names of the columns match the supported names. For example:

```
Login ID, First Name, Last Name, Email
Planning Analytics/Mary Smith,Mary,Smith,msmith@ca.ibm.com
Planning Analytics/Robert Smith,Robert,Smith,roberts@ca.ibm.com
Planning Analytics/Kevin Alexander,Kevin,Alexander,kalex@ca.ibm.com
```

The CSV file must contain one line for each user with the following information: Login ID, First Name, Last Name, Role, Email, Status. Login ID, First Name, and Last Name are required. Role, Email, and Status are optional.

If you omit Role, it defaults to Analyst. For information about roles, see "User roles" on page 31.

If you omit Email, it defaults to Login ID. If the Login ID is not a valid email address, the user will be added with a state of `Not invited yet` and cannot log in to Planning Analytics Workspace Local.

Status can be `Active`, `Inactive`, or `Inactive_suspended`. If you omit status, it defaults to Suspended. A user that has a status of Suspended can't log in to Planning Analytics Workspace until you activate them.

Note: Text values in the CSV file such as role and status must be in English.

If the CSV file does not have a header row, use commas to ensure that the information is in the correct layout. For example:

```
Planning Analytics/Mary Smith,Mary,Smith,,msmith@ca.ibm.com,Active
Planning Analytics/Robert Smith,Robert,Smith,,roberts@ca.ibm.com,Suspended
Planning Analytics/Kevin Alexander,Kevin,Alexander,Administrator,kalex@ca.ibm.com,Active
```

2. On the **Welcome** page, click your user name, and then click **Administer**.

   Note: If you don't see the **Administer** option, then you are not logged in as administrator and you cannot add users in Planning Analytics Workspace Local.

3. If you are not already there, click the **Users** tab.
4. Click **Upload users**.
5. Drag the CSV file onto the **.csv** icon, or browse to the file location and import the file.

   The file is uploaded and you see a success message.

6. Click **OK**.

   The users appear under the **Users** tab with the status that you specified in your CSV file.

   Note: Status appears as Active, Suspended, Invited, or Not invited yet depending on whether you supplied a Status in your CSV file.

Promote a user to administrator (local only)

If your IBM Planning Analytics Workspace administrator is no longer available, you can promote a non-admin user to the administrator role by using the command line.

About this task

This approach is the only way to reset the administrator of Planning Analytics Workspace if the administrator is no longer available.

Procedure

1. Run the following command to see a list of user IDs in this environment and their roles.

   - On Windows:

     ```
     docker exec -it wa-proxy powershell -command "C:\wa-proxy\tools\runTool.ps1"
     ```

   - On Linux:

```
docker exec -it wa-proxy bash -C /wa-proxy/tools/runTool.sh
```

2. Select the user ID that you want to promote and rerun the same command with that user ID:
   - On Windows:

     ```
     docker exec -it wa-proxy powershell -command "C:\wa-proxy\tools\runTool.ps1" <userId>
     ```

   - On Linux:

     ```
     docker exec -it wa-proxy bash -C /wa-proxy/tools/runTool.sh <userId>
     ```

Activate or deactivate a user (local only)

Users in IBM Planning Analytics Workspace Local can be activated or deactivated.

Before you begin

To administer users in IBM Planning Analytics Workspace Local, you must have the **Administrator** role.

Procedure

1. Set the ENABLE_USER_IMPORT configuration parameter to false.

   Note: When you change the paw configuration file, you must run `./scripts/`paw for your operating system or click **Start** in the Planning Analytics Workspace administration tool for your changes to take effect. Only services that are affected by the configuration change will restart.

   For more information, see Planning Analytics Workspace configuration parameters (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_config_file.html).
2. On the **Welcome** page, click your user name, and then click **Administer**.

   Note: If you don't see the **Administer** option, then you are not logged in as administrator and you cannot activate or deactivate users.
3. To invite users that you uploaded with the ADD directive, select the users with a status of **Not invited yet**. Click the circle next to a user name to select an individual user. Click the circle ○ at the top of the list to select all users. To select multiple adjacent users, click the circle next to the first user and then press Shift + click the circle next to the last user.

   A user is selected when a check mark in the circle ✓ accompanying the user name.
4. Beside **Applicable actions**, click **Activate/Deactivate**.
5. In the **Select a state** window, choose **Activate** or **Deactivate**.

   The user's status is changed.
6. Optional: In the Users list, click an individual user profile and click **Activate** or **Deactivate** to toggle the user's state.

Delete a user (local only)

Users in IBM Planning Analytics Workspace Local can be deleted.

Before you begin

To administer users in IBM Planning Analytics Workspace Local, you must have the **Administrator** role.

Procedure

1. Set the ENABLE_USER_IMPORT configuration parameter to false.

   Note: When you change the paw configuration file, you must run `./scripts/`paw for your operating system or click **Start** in the Planning Analytics Workspace administration tool for your changes to take effect. Only services that are affected by the configuration change will restart.

For more information, see Planning Analytics Workspace configuration parameters (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_config_file.html).

2. On the **Welcome** page, click your user name, and then click **Administer**.

   Note: If you don't see the **Administer** option, then you are not logged in as administrator and you cannot delete users.

3. To delete users that you uploaded with the REMOVE directive, select the users with a status of **Not invited yet**.

   Click the circle next to a user name to select an individual user. Click the blue circle at the top of the list to select all users. To select multiple adjacent users, click the circle next to the first user and then press Shift + click on the circle next to the last user. A user is selected when a check mark appears in the circle accompanying the user name.

4. Beside **Applicable actions**, click **Delete users**.
5. In the **Confirm deletion** window, click **OK** to confirm the deletion or click **Cancel** to keep these users.

   A success message appears.

6. Click **OK**.

   The user is deleted.

Delete multiple users (local only)

Multiple users in IBM Planning Analytics Workspace Local can be deleted in bulk with a .csv file.

Before you begin

To administer users in IBM Planning Analytics Workspace Local, you must have the **Administrator** role.

To delete multiple users from Planning Analytics Local, create a CSV file that contains one line for each user that you want to delete with the following information: Login ID, first name, last name, role, environment. Use the `REMOVE` directive.

Note: Text values in the CSV file, including column titles, roles, and environments, must be in English.

You can also export users to a CSV file and edit the file in a text editor to contain one line for each user that you want to delete. Change the `ADD` directive to `REMOVE`.

Procedure

1. Set the ENABLE_USER_IMPORT configuration parameter to false.

   Note: When you change the paw configuration file, you must run `./scripts/paw` for your operating system or click **Start** in the Planning Analytics Workspace administration tool for your changes to take effect. Only services that are affected by the configuration change will restart.

   For more information, see Planning Analytics Workspace configuration parameters (https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_inst.2.0.0.doc/c_paw_config_file.html).

2. On the **Welcome** page, click your user name, and then click **Administer**.

   Note: If you don't see the **Administer** option, then you are not logged in as administrator and you cannot delete users.

3. Click the **Users** tab.
4. Click **Upload users**.
5. Drag the CSV file on to the **.csv** icon .csv, or browse to the file location and import the file.

   The file is uploaded and you see a success message.

6. Click **OK**.

Administer groups

Groups can be used to help manage and organize large numbers of users. Administrators can create, modify, or delete groups by using CSV files.

About this task

To create multiple groups, delete existing groups, or to add users to existing groups, create a CSV file and click **Upload groups**. To create one group at a time, click **Create** and enter the required information. You can also click **Export groups** to save and edit a correctly formatted CSV file that you can use to make changes to existing groups.

You can view this video to learn how to manage groups in Planning Analytics Workspace Classic.

https://youtu.be/8pLNuX5QvIY

Administer groups in Planning Analytics Workspace

Procedure

1. To create one group at a time:
   a) On the Home page, click the **Administration** tile.

      Note: You can't create a group with the name `Everyone` because Planning Analytics Workspace uses that as a default group when there are no groups defined.

   b) Click **Users & Groups**.

   c) Click **Groups**.

   d) Click **Create a new group** +.

   e) Enter a **Name** and **Description** for the group, then click **Submit**.

      The new group opens on the **Users & Groups** page.

   f) Click **Manage**.

   g) Select the users you want to add to the group, then click **Save**.
2. To create multiple groups, delete existing groups, or to add users to existing groups, create a CSV file that contains the groups to create or update.

   The format of the CSV file is described in the user interface. Use a header row as the first row of your CSV file. The CSV file must contain one line for each action with the following information: Group, Login ID, and Directive (ADD or REMOVE). The Login ID must be a valid user in this environment.

   The following example .csv file adds four groups (`Test`, `Development`, `Quality`, `Management`) and adds the Login ID `admin` to each group.

   ```
   Group,Login Id,Directive
   Test,admin,ADD
   Development,admin,ADD
   Quality,admin,ADD
   Management,admin,ADD
   ```

   Tip: You can leave the Login ID blank (with no spaces) to create an empty group with no users in it.

   a) On the Home page, click the **Administration** tile.

   b) Click **Users & Groups**.

   c) Click **Groups**.

   d) Click **Upload groups from a csv file**.

   e) Drag and drop your .csv file onto the **Drop your .csv file here** region or click the region and browse for your file.

Administer groups Planning Analytics Workspace Classic

Procedure

1. To create one group at a time, click **Create**. Enter a **Name** and a **Description** and select one or more users to add to the group.

   Note: You can't create a group with the name `Everyone` because Planning Analytics Workspace uses that as a default group when there are no groups defined.
2. To create multiple groups, delete existing groups, or to add users to existing groups, create a CSV file that contains the groups to create or update.

   The format of the CSV file is described in the user interface. Use a header row as the first row of your CSV file. The CSV file must contain one line for each action with the following information: Group, Login ID, and Directive (ADD or REMOVE). The Login ID must be a valid user in this environment.

   The following CSV file adds four groups (`Test`, `Development`, `Quality`, `Management`) and adds the Login ID `admin` to each group.

   ```
   Group,Login Id,Directive
   Test,admin,ADD
   Development,admin,ADD
   Quality,admin,ADD
   Management,admin,ADD
   ```

   Tip: You can leave the Login ID blank (with no spaces) to create an empty group with no users in it.
3. On the **Welcome** page, click your username, and then click **Administer**.
4. Click the **Groups** tab.
5. Click **Upload groups**.
6. Drag the CSV file on to the **.csv** icon, or browse to the file location and import the file.

   The file is uploaded and you see a success message. Click **OK**.

Export users

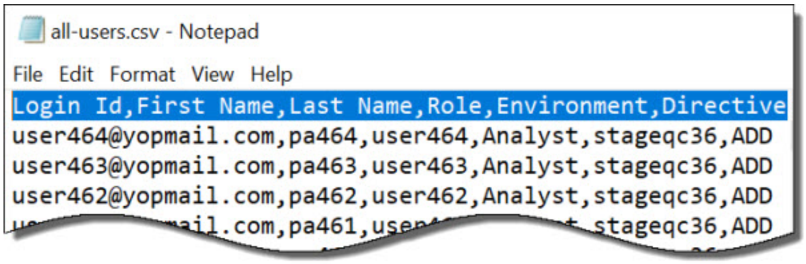
If you are an administrator, you can export all users in your environment to a CSV file. The exported CSV file can be updated and imported back into IBM Planning Analytics Workspace, which makes it easy to update roles and environments of users with a single bulk upload.

About this task

You can export users in a CSV file and save the file to your Downloads folder as `all-users.csv` or open it in the editor and save it with a file name of your choice.

The CSV file is formatted so that you can edit it and upload users back into your environment with any changes that you made. Columns in the exported CSV file match the format that is required for importing users. The `Directive` is hardcoded as ADD, which can be used to update existing users.

Each row in the CSV file represents a user in an environment. If a user is in multiple environments, there will be multiple rows for that user.



```
Login Id,First Name,Last Name,Role,Environment,Directive
user464@yopmail.com,pa464,user464,Analyst,stageqc36,ADD
user463@yopmail.com,pa463,user463,Analyst,stageqc36,ADD
user462@yopmail.com,pa462,user462,Analyst,stageqc36,ADD
ail.com,pa461,use stageqc36,ADD
```

Export users in Planning Analytics Workspace

Procedure

1. On the Home page, click the **Adminitration** tile.
2. Click **Users & Groups**.
3. Click the **Users** tab.

   Note: Selecting users has no impact. Clicking **Download users** always exports all users in the current environment.

4. Click **Download users** ⤓.

Export users in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the **Users** tab.

   Note: Selecting users has no impact. Clicking **Export users** always exports all users in the current environment.
3. Click **Export users**.
   - Select **Open** to open the exported CSV file and save it to a location with your choice of file name.
   - Select **Save** to save the exported CSV file to your Downloads folder as `all-users.csv`.

Change a user's role

You can change the role for one or more users. In the user interface, you can select a single user and change their role. Or, you can export users to a CSV file, make changes to roles and then upload those changes.

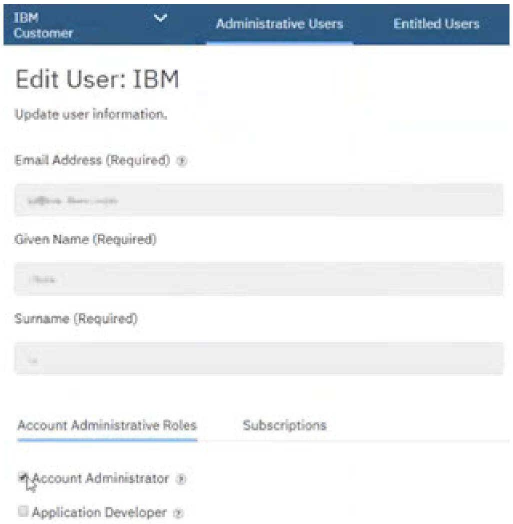
Note: In a user's profile, you can see only the role that a user has in the current environment. If they belong to another environment with a different role, you must log in that environment to see that role reflected in the user profile.

When you make changes to the roles of users in the current environment using a CSV upload, you will see the changes reflected in the users' profiles in the current environment.

If you make changes to the roles of users in other environments using a CSV upload, you must log in to those environments to see the changes reflected in the users' profiles.

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

☑ Account Administrator

□ Application Developer

About this task

The following roles are available:

Consumer

Consumers can open books and views and other content that is shared with them.

Consumers cannot create their own books and views, but they can share content that is shared with them, with consumer rights only.

Consumers can delete books and views if they have **Full control** permission set for those books and views.

Note: When you log in to Planning Analytics Workspace from an iPad, you are always a Consumer, regardless of your actual role. For more information, see "Accessing Planning Analytics Workspace from Apple iPad" on page 36.

Analyst

Analysts have the rights of a consumer, plus the following abilities:

Analysts can create, edit, and share books and views.

Analysts can delete books and views if they have **Full control** permission set for those books and views.

Modeler

Modelers have the rights of an analyst plus the following abilities:

They can create and share content, and edit content that is shared with them.

Modelers can create, edit, and delete dimensions and hierarchies. They can add, delete, cut, paste, copy, move, and sort members and their attributes in a hierarchy.

For more information, see "The modeler role" on page 184.

Administrator

Administrators have all the rights of a modeler, plus the following abilities:

Administrators can see all content in the workspace.

Administrators can set permissions for a book.

Note: At least one user in your organization must have the administrator role. If your organization has only one administrator, this user cannot be deleted or assigned to another role. Administrators can assign roles to users and only an administrator can change another user's role to administrator.

By default, the administrator role is assigned to the first user in your organization's Planning Analytics Workspace account. In Planning Analytics Workspace on cloud, this first administrator is also known as the *subscription administrator*. Only the subscription administrator can add new users.

Change a user's role in Planning Analytics Workspace

Procedure

1. On the Home page, click the **Administration** tile.
2. Click **Users & Groups**.
3. Click the **Users** tab.
4. To change the role for a single user:
   a) Select the user on the **Users** tab.
   b) Select a new role on the **Details** tab.
   c) Click **Apply**.
5. To change multiple users to the same role:
   a) Select the users on the **Users** tab.
   b) On the selected items pane, click **Set role**.
   c) Select a role for the users and click **Apply**.

Change a user's role in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the circle next to a user name to select an individual user. Click the circle ○ at the top of the list to select all users. To select multiple adjacent users, click the circle next to the first user and then press Shift + click the circle next to the last user.

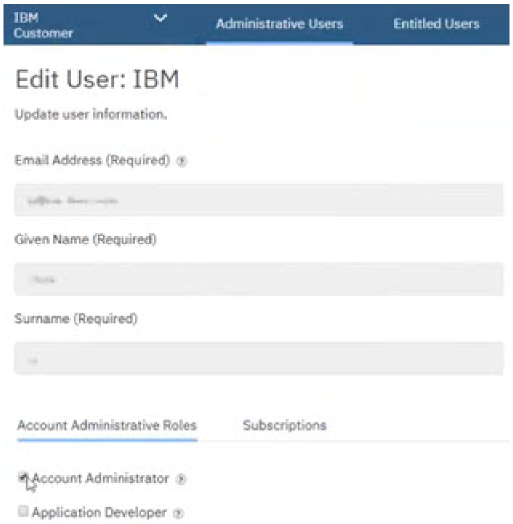
   A user is selected when a check mark in the circle ✓ accompanying the user name.
3. Beside **Applicable actions**, click **Manage roles**.
4. In the **Select a role** window, select a role.

Change a user's environment

A user's environment may need to be changed as their work needs evolve or change over time. An administrator can use a CSV file or the user interface to change a user's environment.

Before you begin

- You must be a member of the same organization as the users you are inviting or uploading.
- You must be an account administrator for your organization, as defined in the Subscription and Subscriber Management tool.



IBM Customer

Administrative Users

Entitled Users

Edit User: IBM

Update user information.

Email Address (Required)

Given Name (Required)

Surname (Required)

Account Administrative Roles

Subscriptions

☑ Account Administrator

☐ Application Developer

About this task

Users in IBM Planning Analytics Workspace can have access to one or more environments. For example, a user can access both your non-production and production environments. In the user interface, you can select a single user and change their environment. Or, you can export users to a CSV file, make changes to environments and then upload those changes.

You can change the environment for one or more users at time.

Results

If a user has multiple environments assigned to it, they will be prompted to select one the next time they log in.

Change a user's environment in Planning Analytics Workspace

Procedure

1. On the Home page, click the **Administrator** tile.
2. Click **Users & Groups**.
3. Click the **Users** tab.
4. To change environments for a single user:
    a) Select the user on the **Users** tab.
    b) On the adjacent user detail pane, click the **Environments** tab.
    c) Click **Manage**.
    d) Select the environments for the user, then click **Save**.
5. To change environments for multiple users:
    a) Select the users on the **Users** tab.
    b) On the selected items pane, click **Set environment**.
    c) Select the environments for the users and click **Save**.

Change a user's environment in Planning Analytics Workspace Classic

Procedure

1. On the **Welcome** page, click your user name, and then click **Administer**.
2. Click the **Users** tab.

3. Click the circle next to a user name to select an individual user. Click the blue circle at the top of the list to select all users.

   To select multiple adjacent users, click the circle next to the first user and then press Shift + click on the circle next to the last user. A user is selected when a check mark appears in the circle accompanying the user name.
4. Beside **Applicable actions**, click **Manage environments**.

   Note: If you don't see **Manage environments**, then you've selected a user that doesn't have a **Status** of **Active**.
5. In the **Select environments** window, select one or more environments.

   The user's primary environment is displayed in bold text and can't be changed. You can select only environments that you are an administrator for.

   A light blue check mark ✓ beside an environment indicates that some of the selected users already have access to it. If you select this environment all selected users will have access to it.

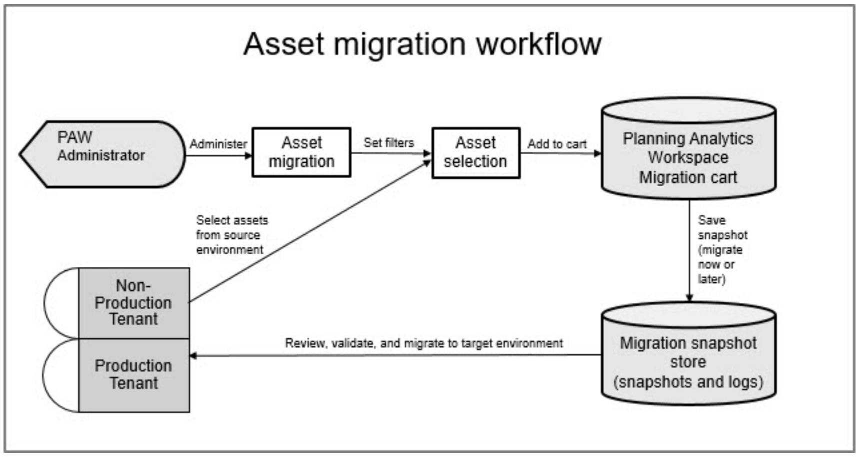
   A dark blue check mark ✓ beside an environment indicates that the user is already a member of this environment. Selecting this environment removes the user from the environment. You can **Cancel** or **Continue**.
6. Close the **Select environments** window.

Migrate assets with Lifecycle Management

Assets such as books and views can be migrated by an administrator from a source environment to target environment using Lifecycle Management.

If you are an administrator, you can migrate any assets that you have permission to view from a source environment to a target environment by clicking **Lifecycle Management** on the **Administration** page in IBM Planning Analytics Workspace. An asset can be a book or a view. Websheets are not supported as an asset type.

Customers on cloud are provisioned with multiple products in multiple environments. They can have a development environment and a production environment. They might also have a test or sandbox environment, or personal development environments.


Asset migration workflow
PAW Administrator
Administer
Asset migration
Set filters
Asset selection
Add to cart
Planning Analytics Workspace Migration cart
Select assets from source environment
Save snapshot (migrate now or later)
Non-Production Tenant
Production Tenant
Review, validate, and migrate to target environment
Migration snapshot store (snapshots and logs)

As an administrator, you can migrate any assets that you have permission to view from one environment to another environment. This capability makes it easier to build content in a development environment and then migrate it to your production environment when you are ready. If the target environment is available, you can migrate assets in one step. If the target environment is a IBM Planning Analytics Local instance, you can export your assets from your source environment and import them in the target environment.

Note: You must have at least **View** permission for an asset to see it in Lifecycle Management. You are not shown assets that you don't have permission to view even if they exist in your source environment.

You can also migrate assets from a TM1 database to another TM1 database by migrating assets to your source environment and selecting a new TM1 database. Or, you can create snapshots of assets as a backup so that you can restore an earlier version of your assets whenever you want.

Remember: A snapshot is a collection of assets at a single point in time. Lifecycle Management validates that the content of the snapshot that you are creating can be migrated to the target environment. When you migrate a collection of assets from the source environment to the target environment, the contents of the assets are migrated. If a book contains a websheet, the websheet must be migrated manually.

When you migrate assets to a target environment, the assets are added to a new folder in your personal space in your target environment. You can choose when to move the assets from your personal folder into a shared folder.

Prerequisites

To migrate assets with Lifecycle Management, you must be logged in to IBM Planning Analytics Workspace as an administrator.

When you are an administrator, the **Administration** tile is available on the Home page. Click the tile to open the Administration page.

For more information about the administrator role, see "User roles" on page 31.

Limitations

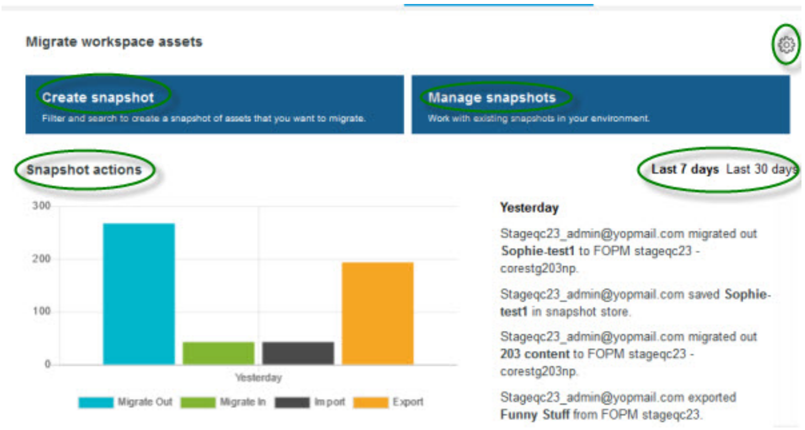
The following limitations apply to Lifecycle Management in IBM Planning Analytics Workspace.

- You must have at least **View** permission for an asset to see it in Lifecycle Management. You are not shown assets that you don't have permission to view even if they exist in your source environment.
- You can migrate assets that exist in the Shared folder or in your Personal folder. You cannot migrate assets that exist in other users' Personal folders.
- Websheets are not supported as an asset type. You cannot create a snapshot that includes a websheet. Websheets must be manually moved to the target environment and must be placed in the same application folder structure.
- If you are migrating a book that contains multiple missing databases, when you remove the book from the snapshot all of the relevant warnings for that book are not removed.
- Migration of a view puts an incorrect **Modified by** user in the collection even though the **Modified by** time is updated.
- If you create a circular mapping of databases, all databases are mapped to a single database. For example, mapping 24Retail to 2003cert and 2003cert to 24Retail maps all databases to 24Retail.
- When you migrate out and then migrate in to the same environment, both actions appear in the **Snapshot actions** chart. However, only the migrate out action appears in the log.
- When you migrate a previously saved snapshot, you are able to select a folder name in the target environment. When you migrate a new snapshot, you must use the snapshot name as the folder name.
- When you validate a snapshot, the validation can see only actively running TM1 databases. You can migrate a snapshot that contains validation errors if you know that the target database will be running soon.

Get started

On the **Home** page, click the **Administration** tile to open the **Administration** page. To migrate assets, click **Lifecycle Management** on the **Administration** page.

The home page of **Lifecycle Management** has two actions tiles, a graphical summary of snapshot actions, and a condensed list of snapshot actions. There is also a settings button that you can use to define database mappings for your source and target environments.



Filter and search to create a snapshot of assets that you want to migrate.

Work with existing snapshots in your environment.

Yesterday

Stageqc23_admin@yopmail.com migrated out **Sophie-test1** to FOPM stageqc23 - corestg203np.

Stageqc23_admin@yopmail.com saved **Sophie-test1** in snapshot store.

Stageqc23_admin@yopmail.com migrated out **203 content** to FOPM stageqc23 - corestg203np.

Stageqc23_admin@yopmail.com exported **Funny Stuff** from FOPM stageqc23.

You can use the actions tiles to create a snapshot or manage existing snapshots. Click one of these tiles to get started.

The **Snapshot actions** pane displays a stacked bar chart of the Lifecycle Management actions by date, showing the most recent actions first. The chart shows the number of assets that were migrated out, migrated in, imported, or exported over time. By default, the **Last 7 days** option is selected, showing migration actions within the most recent 7 days on the horizontal axis. To see the last 30 days of actions, click **Last 30 days**.

Note: When you select **Last 7 days** (default), the chart shows days in the horizontal axis. When you select **Last 30 days**, the chart shows weeks on the horizontal axis. If no migration actions were performed on a day or week, the chart does not show that day or week on the horizontal axis.

The number of assets is shown on the vertical axis. Each bar in the chart represents a snapshot and the bar is colored to indicate what action happened on that snapshot. If you hover over a bar in the chart, a text bubble shows the number of assets in the snapshot. You can glance at the chart, see peaks when large numbers of assets were migrated, and determine which actions occur most frequently.

Beside the **Snapshot actions** pane, you can see a quick list of recent Lifecycle Management actions for the last seven days or the last 30 days. You can see who saved, migrated, exported, or imported a snapshot and where they performed that action.

You can think of the following steps as the general work flow of Lifecycle Management:

1. “Create a snapshot of assets” on page 296
   - Filter and select books and views.
   - Add assets to your cart.
   - Optionally, review your cart, setting the target folder location and the TM1 database.
   - Optionally, save assets in a snapshot, adding a snapshot name and description, and migrate it later.
   - Optionally, save assets in a snapshot and export the snapshot to your file system in one step.
   - Migrate your assets to the target environment.
2. “Manage snapshots” on page 303
   - Find existing snapshots in your source environment.
   - Import a snapshot from your local file system.
   - Rename or delete an existing snapshot.
   - View the logs for an existing snapshot.
3. Configure your database mappings.
   - Create a mapping for a database in the source environment to a database in the target environment.
   - Modify an existing mapping.

Create a snapshot of assets

You can add multiple assets to a snapshot so that you can migrate the assets to a target environment.

Procedure

1. Click **Create snapshot** from the home page of **Lifecycle Management** to get started migrating assets.

   The **Create snapshot** page opens. Your source environment defaults to your current environment.
2. "Filter to find assets" on page 296 and then migrate your assets to a target environment.

   You must apply your filters to select assets to include in your snapshot.
3. Click **Home** to return to the **Lifecycle Management** home page.

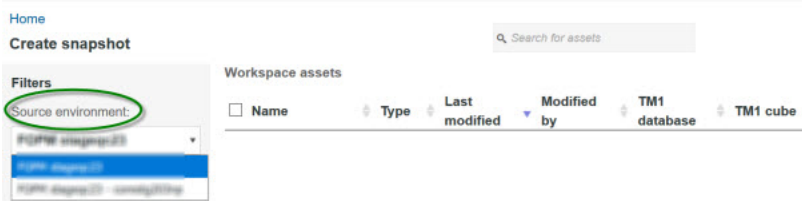
Filter to find assets

When you create a snapshot, you can use filters to narrow the selection of assets to include in your snapshot.

When you click **Create snapshot**, you can use filters to find assets to add to your snapshot. You must apply your filters to select assets to include in your snapshot. Only assets that are available to you are shown. You can add assets that exist either in the Shared folder or your Personal folder. You cannot add assets that exist in other users' Personal folders.

Source environment

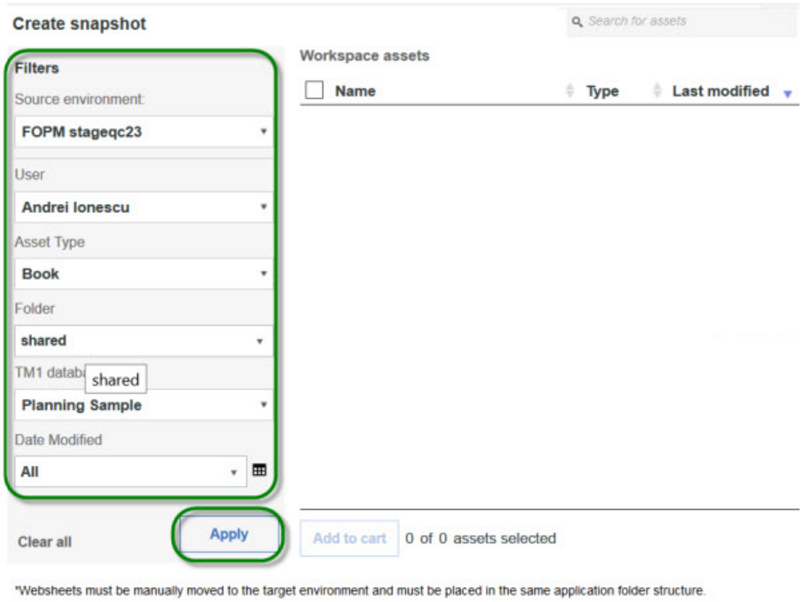
You can select a source environment to use to find assets that are available in IBM Planning Analytics Workspace. By default, the source environment is the environment that you are logged in to. Only available environments are shown in the **Source environment** drop-down list.



When you select an environment from the **Source environment** drop-down list, you can filter or search for assets that are available in that environment

Filters

You can use filters to narrow the selection of assets to include in your snapshot.



Create snapshot

Search for assets

Filters

Source environment: FOPM stageqc23

User Andrei Ionescu

Asset Type Book

Folder shared

TM1 datab shared Planning Sample

Date Modified All

Clear all | Apply

Workspace assets

Name	Type	Last modified

Add to cart 0 of 0 assets selected

*Websheets must be manually moved to the target environment and must be placed in the same application folder structure.

You can use any of the following filters:

- **User** You can filter assets by the user that created the asset. The drop-down list is populated automatically with users who have assets that are associated with them when you log in to Planning Analytics Workspace. As an administrator, you can select assets that are owned by all users that appear in the list.
- **Asset Type** You can filter assets by type, such as **Book**. Currently, the supported asset types include **Book** and **View**.
- **Folder** You can filter assets by whether they are in the **Shared** or your **Personal** folder in the source environment.

  You can expand ⊞, collapse ⊟, and scroll through all the subfolders below the **Shared** or **Personal** folder. If there are no subfolders below a folder, you see the simple folder icon 🗀. When you find the folder that you want to select, click the name of the folder.

  When you click **Apply**, filtering finds assets in that folder and in all its subfolders.
- **TM1 database** You can filter assets by which TM1 database they belong to in the source environment. The drop-down list is automatically populated with TM1 databases that have assets that are associated with them. You can also filter assets that are not in any TM1 database by selecting **None** from the drop-down list.
- **Date Modified** You can filter assets by the date when they were modified. You can filter assets that were modified today, yesterday, within the last 7 days, within the last 30 days, or within the last 365 days. You can select a specific date or select a date range by using the data picker.

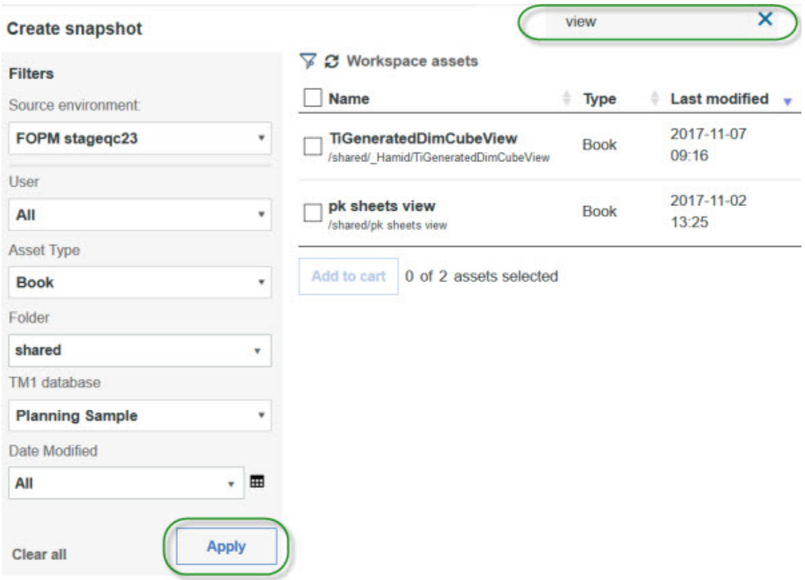
By default, all filters are set to **All** to include all assets.

1. To apply the filters, click **Apply**.

   When you change your selection in the **Filters** pane, the assets in the **Workspace assets** pane change to reflect the filter only when you click **Apply**.
2. To clear your filter selections and start over, click **Clear all**.
3. To toggle whether the filters appear, click ▽.

Search for assets

You can filter for specific assets in the **Workspace assets** pane by using the **Search for assets** text field. This filter finds all assets that match all or part of the criteria in any fields of the assets. This filter shows the complete set of assets that match your criteria. This makes it easy to search for assets that match your criteria, select all of the matching assets, and then add them to your cart.



Create snapshot

view ✕

Filters

Source environment: FOPM stageqc23

User
All

Asset Type: Book

Folder: shared

TM1 database: Planning Sample

Date Modified: All

Clear all | Apply

Workspace assets

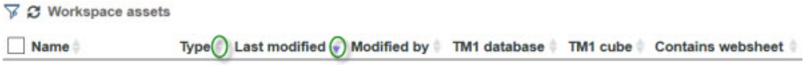
□ Name	Type	Last modified
□ TiGeneratedDimCubeView /shared/_Hamid/TiGeneratedDimCubeView	Book	2017-11-07 09:16
□ pk sheets view /shared/pk sheets view	Book	2017-11-02 13:25

Add to cart 0 of 2 assets selected

1. Type your criteria in the **Search for assets** text field.
2. Click **Apply**.

Sort

You can sort your assets in the **Workspace assets** pane.



Workspace assets

□ Name Type Last modified Modified by TM1 database TM1 cube Contains websheet

1. To sort by any field, click ⬍ beside the field that you want to sort on.

   The icon indicates which field is sorted on and in which direction. For example, if the **Workspace assets** pane shows Name ▲, the sort icon beside **Name** is in focus and it indicates that you are sorting the assets alphabetically in ascending (A to Z) order.

2. To sort the assets alphabetically in descending (Z to A) order, click ▲ beside **Name**.

   Tip: You can sort on only one column at a time.

What to do next

When you are happy with your asset selections, continue to the next step to add assets to your cart.

Note: If you change your filters or add search criteria that changes the assets in your **Workspace assets** pane **after** you select assets to add to your cart, a dialog box tells you that your selection will be cleared. You can add the selected assets to your cart or clear the current selection. It is a good practice to apply filters and search criteria **first** and then select assets.

Add assets to your cart

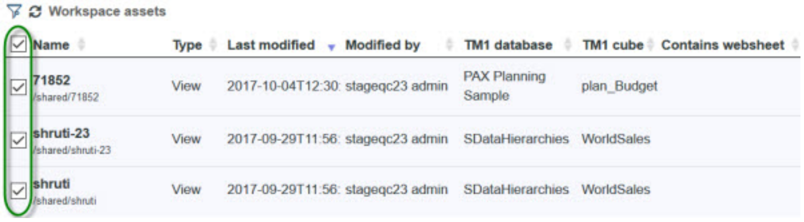
When you are satisfied with your filtered assets, select assets and click **Add to cart** to add assets to a snapshot. You must select at least one asset to add to your snapshot.

About this task

Remember: A snapshot is a collection of assets at a single point in time.

Procedure

1. Select individual assets by clicking the check box beside an asset in the **Workspace assets** pane. Clear the check box beside an asset if you don't want to add it to your cart.
2. Click the check box on the heading row to select all of the assets in your **Workspace assets** pane.

   Workspace assets

☑	Name	Type	Last modified	Modified by	TM1 database	TM1 cube	Contains websheet
☑	71852 /shared/71852	View	2017-10-04T12:30:	stageqc23 admin	PAX Planning Sample	plan_Budget	
☑	shruti-23 /shared/shruti-23	View	2017-09-29T11:56:	stageqc23 admin	SDataHierarchies	WorldSales	
☑	shruti /shared/shruti	View	2017-09-29T11:56:	stageqc23 admin	SDataHierarchies	WorldSales	

   Tip: If you change your filters or add search criteria that changes the assets in your **Workspace assets** pane **after** you select assets to add to your cart, a dialog box tells you that you have assets that are selected and your selection will be cleared. You can click **OK** to apply filters and clear your selection or you can click **Cancel** to keep your selection. It is a good practice to apply filters and search criteria **first** and then select assets.
3. When you are ready, click **Add to cart**.

   The **Cart assets** pane reflects the current contents of your snapshot.

   Note:

   - You can put only 100 assets in to your cart. If you exceed this limit, a dialog box tells you that your cart is too large. You must remove some assets and click **Add to cart** again.
   - You can't add assets with the same name to your cart. If you add more than one asset with the same name, a dialog box tells you that your cart cannot contain assets with the same name. Click **OK**, remove some selections, and click **Add to cart**.
   - If you are adding an asset to your cart that contains a websheet, you will see an indicator in the **Contains websheet** column of the **Workspace assets** pane. You can add the asset to your snapshot but you must manually move the websheet to the application folder in the target environment of the asset.
4. When you want to hide the assets that you have already added to your cart, click **Hide assets in cart**.

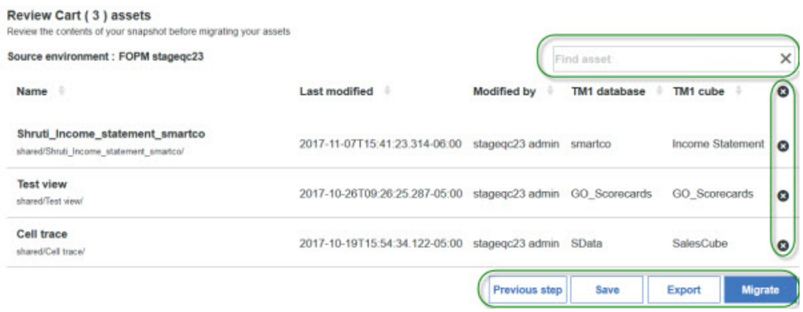
   The assets that you added to your cart appear in your **Cart assets** pane but do not appear in your **Workspace assets** pane. This is a good practice when you use different filter criteria to find assets to add to your cart.
5. Optional: If you want to verify that your assets will migrate successfully, click **Review cart** to review your snapshot and correct errors before you migrate it.
6. Optional: If you are not ready to migrate your assets yet, you can click **Save** to save your snapshot and migrate it later, or if you are an expert, click **Export** to save and export your snapshot in one step.
7. If you are ready, click **Migrate** to migrate your assets in one step.

Review your cart

When you click **Review cart**, you can review the contents of your snapshot before you migrate your assets. Each asset is shown with its name, source folder, last modified date, modified by user name, TM1 database source, and TM1 cube source.

About this task

From the **Review cart** page, you can remove assets from your cart, click **Save** to save your snapshot and exit, click **Export** to save and export a snapshot, or click **Migrate** to save and migrate a snapshot.



Review Cart (3) assets

Review the contents of your snapshot before migrating your assets

Source environment : FOPM stageqc23

Find asset ✕

Name	Last modified	Modified by	TM1 database	TM1 cube	⊗
Shruti_Income_statement_smartco shared/Shruti_Income_statement_smartco/	2017-11-07T15:41:23.314-06:00	stageqc23 admin	smartco	Income Statement	⊗
Test view shared/Test view/	2017-10-26T09:26:25.287-05:00	stageqc23 admin	GO_Scorecards	GO_Scorecards	⊗
Cell trace shared/Cell trace/	2017-10-19T15:54:34.122-05:00	stageqc23 admin	SData	SalesCube	⊗

Procedure

1. To remove an asset from your cart, click ⊗ beside the asset.
2. To remove all assets from your cart, click ⊗ on the header row.
3. To find a particular asset to remove, use the **Find asset** field. The **Find asset** field on the **Review cart** page highlights all of the assets in your cart that match your criteria. All fields are searched for text that matches your criteria.
4. If you want to start again and add or remove assets from this snapshot, click **Previous step**.

   You can filter for more assets to add to your cart and then review your cart again.

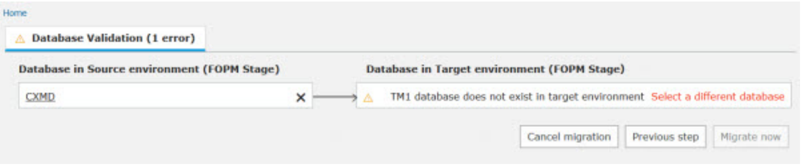
Correct errors and validate

Errors can occur when you migrate your assets. Use the **Database Validation** page to review any validation failures, cancel the migration, correct errors and re-validate, or migrate your assets.

About this task

When you click **Migrate** from the **Review cart** page, you save your snapshot and validate that the TM1 database is available in the target environment. If there are errors, the **Database Validation** page opens. You can review any validation failures, cancel the migration, correct errors and re-validate, or migrate your assets.

The validation page indicates the database in the source environment and the database in the target environment. If the migration of the assets from the source environment database to the target environment database is valid, a ✓ appears beside the database. All migration should appear with a ✓ before you can migrate your snapshot of assets, however you can migrate a snapshot with validation errors. You might migrate a snapshot that has validation errors when the target database is down for maintenance but you know that it will be up soon.



Database in Source environment (FOPM Stage): CXMD

Database in Target environment (FOPM Stage): ⚠ TM1 database does not exist in target environment Select a different database

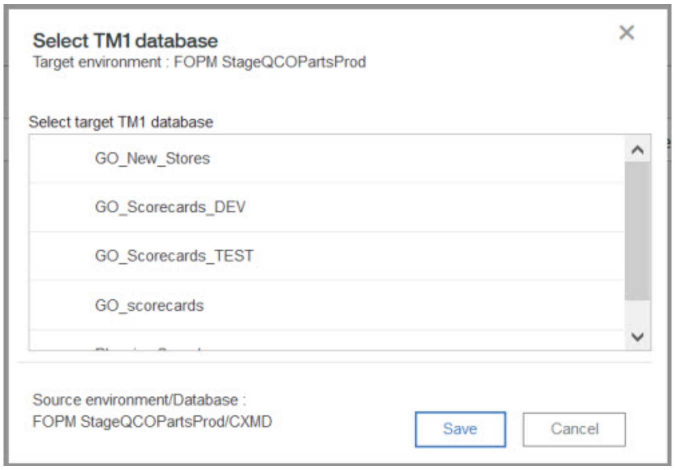
Procedure

1. Click **Cancel migration** to leave the migration process.

   Your snapshot is saved but your assets are not migrated to the target environment. You can revisit you snapshot from the **Manage snapshots** page. For more information, see "Manage snapshots" on page 303.
2. Click **Previous step** to review your cart and correct errors.
3. Correct errors on the **Database Validation** page.

   You can make the following corrections:

- You can remove the database from your snapshot. Click ✕ to remove the asset from the snapshot. All assets that use TM1 content in this database are removed from the snapshot.
- You can change the TM1 database in the target environment and select a replacement TM1 database to resolve a validation error. Click **Select a different database** and select a target TM1 database from the list. Only valid databases that exist in the target environment are shown in the dialog box.



Select TM1 database
Target environment : FOPM StageQCOPartsProd

Select target TM1 database

GO_New_Stores

GO_Scorecards_DEV

GO_Scorecards_TEST

GO_scorecards

Source environment/Database :
FOPM StageQCOPartsProd/CXMD

4. When you have no errors in your validation results, click **Migrate now** to continue to the next step, save and migrate a new snapshot.

Save a snapshot

When you click **Save**, you can save a snapshot of the assets that you have selected.

Procedure

1. Click **Save**.

   The **Save snapshot** dialog box opens.
2. Enter a name for your snapshot so that you can find it easily when you return to it later.

   Note:

   - The name of the snapshot must be unique. If you enter an existing snapshot name, you cannot save your snapshot.
   - The name of the snapshot must not continue invalid characters such as #%&{}\<>*?/$!'":@+` ||=.
   - Leading and trailing blanks are removed from the name of the snapshot.
   - Your snapshot must contain at least one asset.
3. Optional: In the **Description** field, enter a description of the contents of the snapshot.
4. Click **Save**.

What to do next

Click **Manage snapshots** on the **Lifecycle Management** home page to review your snapshot actions.

For more information, see "Manage snapshots" on page 303.

Save and export a snapshot

You can save a new snapshot and export it to your local file system so that you can upload the snapshot into another IBM Planning Analytics Workspace target environment.

Procedure

1. Click **Export** to save and export a snapshot.

   The **Save snapshot to export** dialog box opens.

2. Enter a name for your snapshot so that you can find it easily when you return to it later.

   Note:

   - The name of the snapshot must be unique. If you enter an existing snapshot name, you cannot save your snapshot.
   - The name of the snapshot must not continue invalid characters such as #%&{}\<>*?/$!'":@+` ||=.
   - Leading and trailing blanks are removed from the name of the snapshot.
   - Your snapshot must contain at least one asset.
3. Optional: In the **Description** field, enter a description of the contents of the snapshot.
4. Click **Save and export**.
5. Click **OK** to save the exported file to your file system. The file is saved as a file with the GZ file type (.gz) in your default Downloads folder.

   Tip: If you click **Cancel**, the snapshot is saved and can be accessed by clicking **Manage snapshots** on the **Lifecycle Management** home page but the snapshot is not saved to your file system.

What to do next

Click **Manage snapshots** on the **Lifecycle Management** home page to review your snapshot actions.

For more information, see "Manage snapshots" on page 303.

Save and migrate a new snapshot

When all of the assets in your new snapshot can be migrated successfully, you can click **Migrate**. By default, your assets are migrated to your personal area in a folder with a name that you specify.

Procedure

1. Click **Migrate**.

   The **Save and continue migration** dialog box opens.
2. In the **Select target environment** area, select a target environment to migrate your assets to.

   Only target environments that you have access to are shown. When you select your target environment, a check mark ✓ appears beside the environment name.
3. In the **Snapshot name** field, enter a name for your snapshot of assets.

   Note:

   - The name of the snapshot must be unique. If you enter an existing snapshot name, you cannot save your snapshot.
   - The name of the snapshot must not continue invalid characters such as #%&{}\<>*?/$!'":@+` ||=.
   - Leading and trailing blanks are removed from the name of the snapshot.
   - Your snapshot must contain at least one asset.
4. Optional: In the **Snapshot description** field, enter a description of the contents of the snapshot.
5. Enter the **Folder name** where you want your migrated assets to reside. This folder is created at the top level in your personal folder on Planning Analytics Workspace.
6. To retain the folder structure of the assets in your snapshot, select the **Keep folder structure while deploying** option. When this option is selected, the folder structure of the assets in your snapshot is duplicated in your personal folder when the snapshot is migrated.

   Deselect the **Keep folder structure while deploying** option to migrate all assets into the top level of the folder you specified in step 5.
7. Click **Migrate**.

   - If the migration is successful, a message indicates that your assets were copied successfully to your target environment.

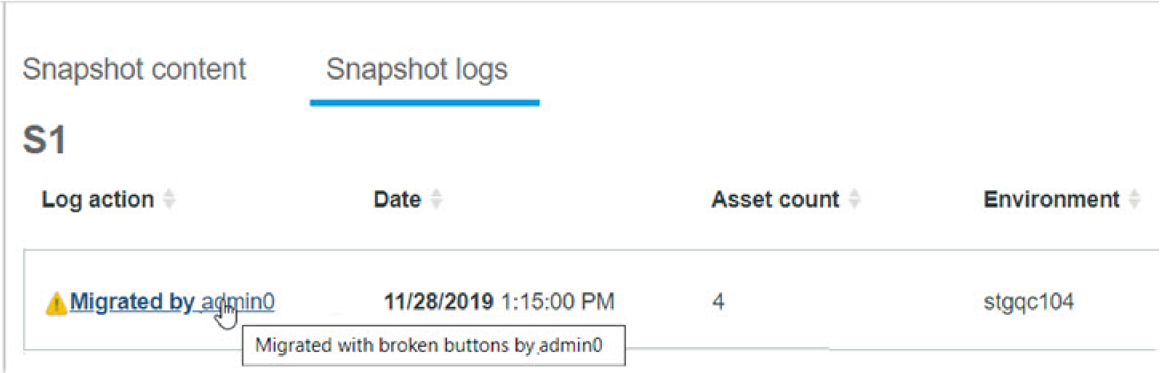
- If the migration isn't successful, the validation page opens and you can correct errors in your snapshot.

Results

If the migration is successful, a message indicates that your assets were copied successfully to your target environment. Additionally, the Snapshot logs tab records the date and time of the migration, the number of assets migrated, and the target environment.

If the migration is successful, but results in broken button links, you can view a list of migrated assets that resulted in broken buttons and take action to restore the buttons in the migration target environment.

When you migrate a snapshot that results in broken buttons, the **Snapshot logs** tab on the **Manage snapshots** page alerts you to this condition. A warning icon next to the **Migrated by** log action indicates the presence of broken buttons in the migrated snapshot. If you hover over the log action, an informational message confirms the issue.

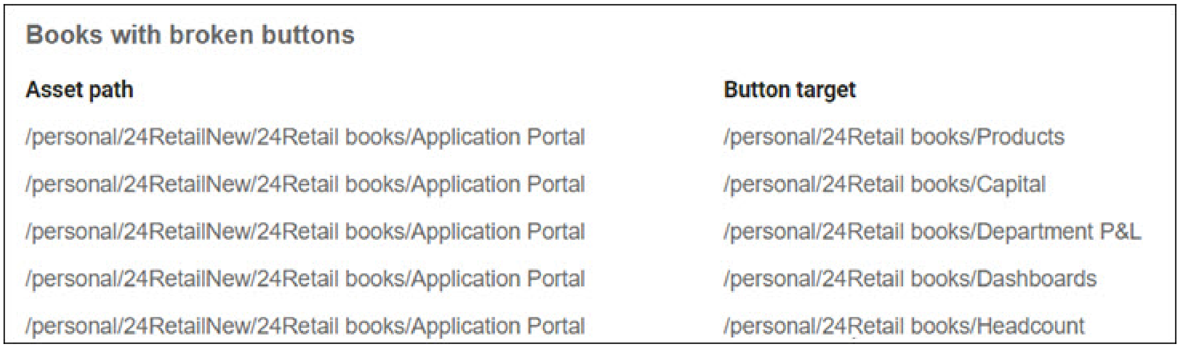


S1

Log action	Date	Asset count	Environment
⚠ Migrated by admin0	11/28/2019 1:15:00 PM	4	stgqc104

Migrated with broken buttons by admin0

When there are broken buttons in a migrated snapshot, the **Migrated by** log action becomes a link. Click the link to view a log report showing assets that contain broken buttons and the missing button targets.



Books with broken buttons

Asset path	Button target
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Products
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Capital
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Department P&L
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Dashboards
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Headcount

You can restore the broken buttons by adding the button targets to your snapshot and repeating the snapshot migration.

If the migration isn't successful, the validation page opens and you can correct errors in your snapshot.

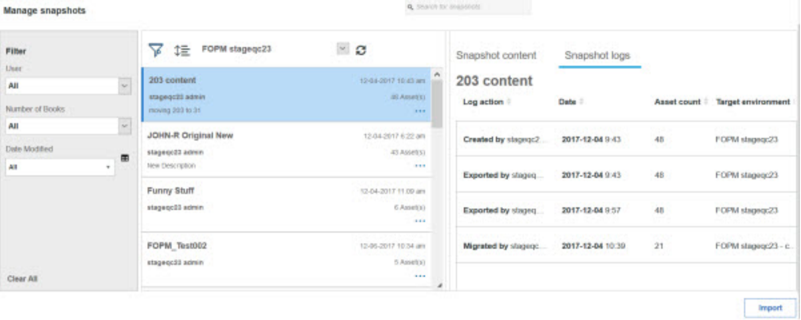
Manage snapshots

You can rename or delete a snapshot, review its contents, and review any logs associated with it.

About this task

From the Lifecycle Management home page, click **Manage snapshots** to view a list of snapshots that have been created. The **Manage snapshots** page opens.

The snapshot list pane shows all snapshots from the current source environment with the current filter and search criteria applied. The results show all snapshots created by all administrators in your current environment. You can manage snapshots that were created by another administrator.



Manage snapshots

Search for snapshots

Filter

User

All

Number of Books

All

Date Modified

All

Clear All

FOPM stageqc23

203 content
stageqc23 admin
moving 203 to 31
12-04-2017 10:43 am
48 Asset(s)

JOHN-R Original New
stageqc23 admin
New Description
12-04-2017 6:22 am
43 Asset(s)

Funny Stuff
stageqc23 admin
12-04-2017 11:09 am
6 Asset(s)

FOPM_Test002
stageqc23 admin
12-06-2017 10:34 am
5 Asset(s)

Snapshot content | Snapshot logs

203 content

Log action	Date	Asset count	Target environment
Created by stageqc2...	2017-12-04 9:43	48	FOPM stageqc23
Exported by stageq...	2017-12-04 9:43	48	FOPM stageqc23
Exported by stageq...	2017-12-04 9:57	48	FOPM stageqc23
Migrated by stageqc...	2017-12-04 10:39	21	FOPM stageqc23 - c...

Import

Procedure

1. Click ⧩ to toggle whether the **Filter** pane appears.

   By default, the **Filter** pane is collapsed.

   The **Filter** pane expands with any filter selections that you entered and any row selections that you made previously. The default is no rows selected. You can change your filter criteria or clear all of your filter criteria. You can search for a snapshot.

   For more information, see "Filter and search for snapshots" on page 304.
2. Click **Import** to import a snapshot from your local file system.

   For more information, see "Import a snapshot" on page 307.
3. Select a snapshot to review its contents in the **Snapshot content** pane.

   The list of snapshots that were created in this environment is shown in the middle pane. Each available snapshot is shown with its name, the name of the user who created it, and its description if it has one. You can also see the date and time that the snapshot was last updated, how many assets it contains, and an action menu of available snapshot options. The **Snapshot content** pane shows the snapshot name, when the snapshot was created, the user that created it, the snapshot description if there is one, and the number of assets that it contains.

   For more information, see "Rename or delete a snapshot" on page 307.
4. Click **Snapshot logs** to view all the actions that were performed on this snapshot.

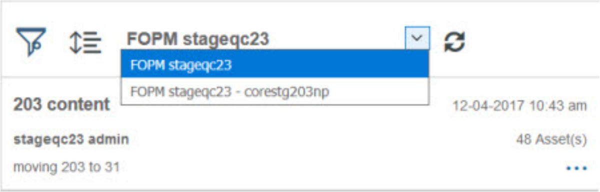
   For more information, see "View snapshot logs" on page 308.

Filter and search for snapshots

You can use the filter and search tools to find an existing snapshot that you saved previously.

Source environment

The **Manage snapshots** page defaults to show you the current source environment. You can select another source environment and see a list of snapshots that were created with assets in the source environment.



FOPM stageqc23

FOPM stageqc23

FOPM stageqc23 - corestg203np

203 content
stageqc23 admin
moving 203 to 31
12-04-2017 10:43 am
48 Asset(s)

Filters

By default, the **Filter** pane is collapsed. To toggle whether the **Filter** pane appears, click ▽. The **Filter** pane expands with any filter selections that you entered and any row selections that you made previously. The default is no rows selected.

You can use the **Filter** pane to limit the snapshots that you see in the snapshot list.

- **User** You can filter snapshots by the user that created the snapshot. The drop-down list is populated automatically when you log in to IBM Planning Analytics Workspace. As an administrator, you can select snapshots that are created by all administrators.
- **Number of assets** You can filter snapshots by the number of assets that they contain. You can find snapshots with any number of assets (All), up to 10 books (0-10), between 11 and 50 books (11-50), or between 51-100 books (51-100).
- **Date Modified** You can filter snapshots by the date that they were modified. You can by Today, Yesterday, Last Week, Last Month, Last Year, or a Custom Range. If you select **Custom Range**, use the date picker to enter your custom date range.

When you change a filter selection, the snapshots in the snapshot list pane change immediately to reflect the filter. To clear your filter selections and start over, click **Clear All**.

Search for snapshots

You can also search the filtered list of snapshots to find a specific snapshot. You can enter search criteria to find snapshots by name or by the name of the person who created them. When you click ⟳, you see an updated list of snapshots in the source environment with the search field emptied and the current row selection cleared. The **Snapshot content** pane is cleared.

Sort

You can sort the list of available snapshots by user, package, date, or number of assets. You can sort ascending or descending.

1. To sort the list of filtered snapshots, click ⇅.
2. To toggle the direction of your sort, click ↑ or ↓.

Snapshot content and actions

When you select a snapshot, you can see a list of assets in the snapshot in the **Snapshot content** pane.

1. To rename or remove a snapshot, click •••. For more information, see "Rename or delete a snapshot" on page 307.
2. To view the logs for a snapshot, click **Snapshot logs**. For more information, see "View snapshot logs" on page 308.
3. To migrate an existing snapshot, click **Migrate**. For more information, see "Migrate an existing snapshot" on page 305.

Migrate an existing snapshot

If you saved a snapshot, you can find that snapshot in the **Manage snapshots** page and migrate it to a target environment.

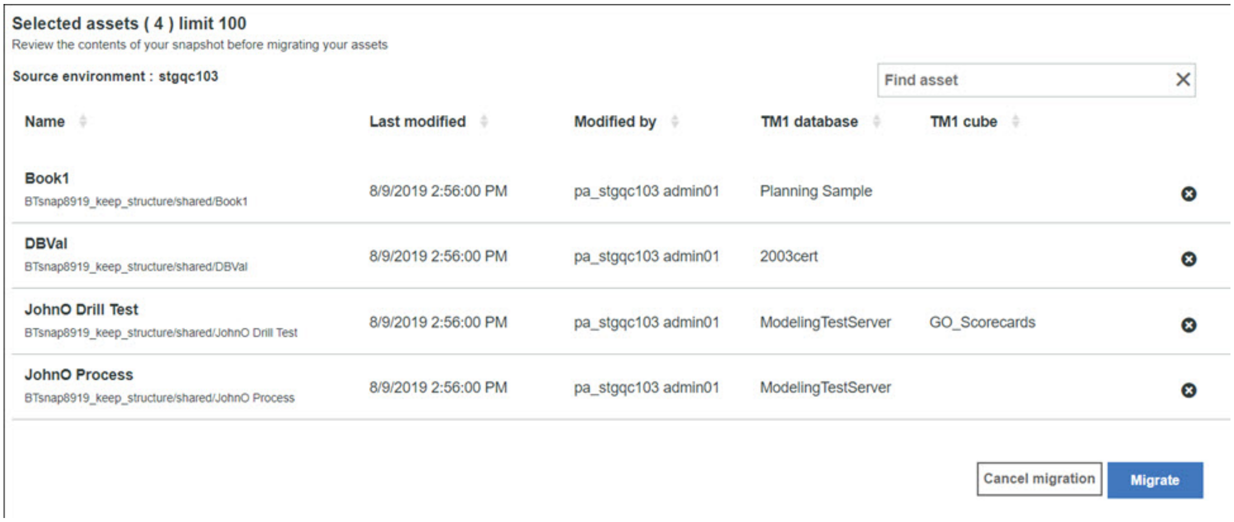
Procedure

1. Click **Manage snapshots** on the **Lifecycle Management** page.
2. On the **Manage snapshots** page, click the snapshot that you want to migrate.

   You can use the options that are described in "Filter and search for snapshots" on page 304 to sort, filter, and search for the snapshot you want to migrate.

3. Click **Migrate**.

   The **Selected assets** page shows you all the assets that are included in the snapshot.



Selected assets (4) limit 100

Review the contents of your snapshot before migrating your assets

Source environment : stgqc103

Name	Last modified	Modified by	TM1 database	TM1 cube	
Book1 BTsnap8919_keep_structure/shared/Book1	8/9/2019 2:56:00 PM	pa_stgqc103 admin01	Planning Sample		⊗
DBVal BTsnap8919_keep_structure/shared/DBVal	8/9/2019 2:56:00 PM	pa_stgqc103 admin01	2003cert		⊗
JohnO Drill Test BTsnap8919_keep_structure/shared/JohnO Drill Test	8/9/2019 2:56:00 PM	pa_stgqc103 admin01	ModelingTestServer	GO_Scorecards	⊗
JohnO Process BTsnap8919_keep_structure/shared/JohnO Process	8/9/2019 2:56:00 PM	pa_stgqc103 admin01	ModelingTestServer		⊗

4. If you want to exclude an asset from the migration, click the remove icon ⊗ for the asset.
5. Click **Migrate** again.

   The **Select target environment** page opens. Use this page to specify the destination for the asset migration and to set other options.
6. Click the target environment where your assets will be migrated.

   Only target environments that you have access to are shown. When you select your target environment, a check mark ✓ appears beside the environment name.
7. Enter the **Folder name** where you want your migrated assets to reside. This folder is created at the top level in your personal folder on Planning Analytics Workspace.
8. To retain the folder structure of the assets in your snapshot, select the **Keep folder structure while deploying** option. When this option is selected, the folder structure of the assets in your snapshot is duplicated in your personal folder when the snapshot is migrated.

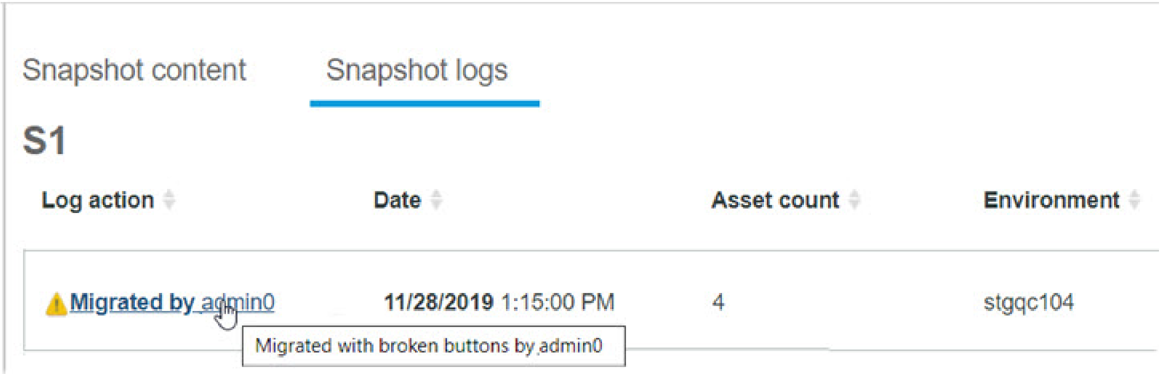
   Deselect the **Keep folder structure while deploying** option to migrate all assets into the top level of the folder you specified in step 7.
9. Click **Migrate**.

Results

If the migration is successful, a message indicates that your assets were copied successfully to your target environment. Additionally, the Snapshot logs tab records the date and time of the migration, the number of assets migrated, and the target environment.

If the migration is successful, but results in broken button links, you can view a list of migrated assets that resulted in broken buttons and take action to restore the buttons in the migration target environment.

When you migrate a snapshot that results in broken buttons, the **Snapshot logs** tab on the **Manage snapshots** page alerts you to this condition. A warning icon next to the **Migrated by** log action indicates the presence of broken buttons in the migrated snapshot. If you hover over the log action, an informational message confirms the issue.

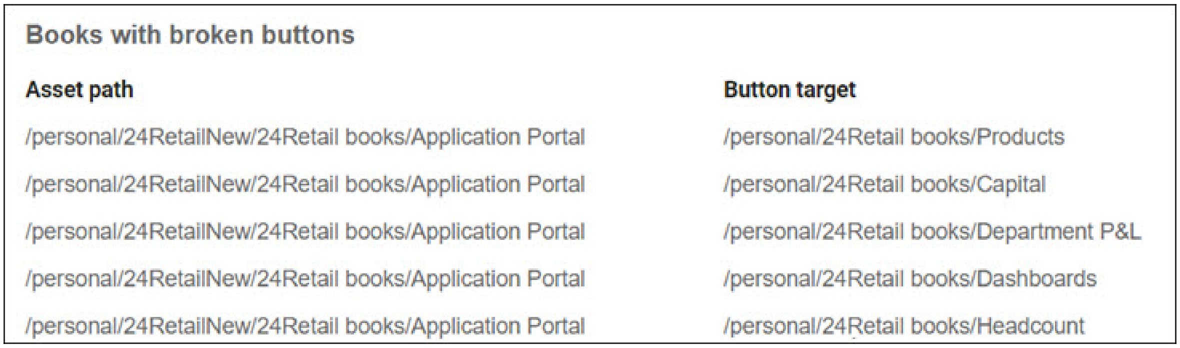


S1

Log action	Date	Asset count	Environment
⚠ Migrated by admin0	11/28/2019 1:15:00 PM	4	stgqc104

Migrated with broken buttons by admin0

When there are broken buttons in a migrated snapshot, the **Migrated by** log action becomes a link. Click the link to view a log report showing assets that contain broken buttons and the missing button targets.



Books with broken buttons

Asset path	Button target
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Products
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Capital
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Department P&L
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Dashboards
/personal/24RetailNew/24Retail books/Application Portal	/personal/24Retail books/Headcount

You can restore the broken buttons by adding the button targets to your snapshot and repeating the snapshot migration.

If the migration isn't successful, the validation page opens and you can correct errors in your snapshot.

Import a snapshot

You can import a snapshot from your local file system into Lifecycle Management.

Procedure

1. Click **Import** to import a snapshot.
2. Select a snapshot from your local file system.

   Note: You must choose a file with the GZ file type (.gz).
3. If a snapshot exists in your source environment with the same name as the file that you are importing, choose one of the following options:

   a) Import and keep both snapshots by renaming the snapshot that is being imported.

   The existing snapshot remains unchanged and both snapshots appear in the snapshot list.

   b) Import and replace the existing snapshot.

   The existing snapshot is deleted. The imported snapshot appears in the snapshot list.

   c) Cancel your import.

   The existing snapshot remains unchanged.

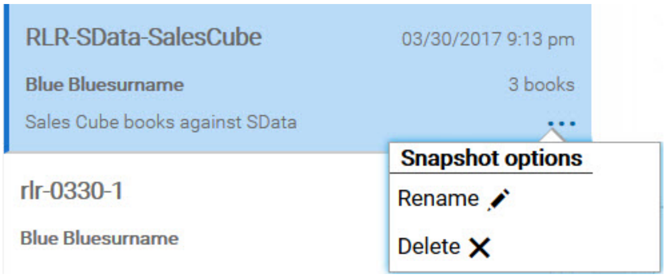
Rename or delete a snapshot

You can change the name and description of a snapshot. You can also delete a snapshot.

Procedure

1. On the **Manage snapshots** page, select a snapshot and click •••.

   The **Snapshot options** action menu opens.


RLR-SData-SalesCube 03/30/2017 9:13 pm

Blue Bluesurname 3 books

Sales Cube books against SData

rlr-0330-1

Blue Bluesurname

2. Click **Rename** on the **Snapshot options** action menu to rename a snapshot.

   You can update the name of the snapshot and the description.

   Note:

   - The name of the snapshot must be unique. If you enter an existing snapshot name, you cannot save your snapshot.
   - The name of the snapshot must not continue invalid characters such as #%&{}\<>*?/$!'":@+` ||=.

3. Click **Delete** on the **Snapshot options** action menu to delete a snapshot a snapshot.

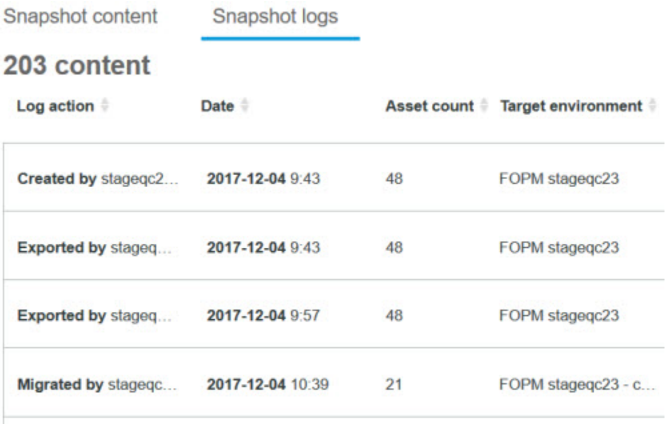
   Note: You must confirm that you want to delete the snapshot. This action cannot be undone.

View snapshot logs

You can view actions for a selected snapshot.

Procedure

1. On the **Manage snapshots** page, select a snapshot.
2. Click **Snapshot logs** to view a list of actions for a snapshot.



203 content

Log action	Date	Asset count	Target environment
Created by stageqc2...	2017-12-04 9:43	48	FOPM stageqc23
Exported by stageq...	2017-12-04 9:43	48	FOPM stageqc23
Exported by stageq...	2017-12-04 9:57	48	FOPM stageqc23
Migrated by stageqc...	2017-12-04 10:39	21	FOPM stageqc23 - c...

Actions are recorded in a log with the following information:

- Snapshot name
- Action that was performed
- User name who performed the action
- Time stamp
- Number of assets in the snapshot
- Target environment for the snapshot

What to do next

Remember: On the Lifecycle Management home page, you can see an activity summary beside the **Snapshot actions** pane. This pane displays a quick list of recent asset migration activities for the last 7 days or the last 30 days. You can see who saved, deployed, exported, or imported a snapshot and where they performed that action.

Configure database mappings

You can configure the mapping of a database in your source environment to a database in your target environment so that you can migrate your assets faster.

About this task

The database mappings streamline common scenarios, such as development to production environment mappings. The mappings are stored for each target environment not for each user. Administrators for the target environment can set up these mappings. You must have permission to view assets in a database in the source and target environments to set up the database mappings.

Note: You can map multiple databases in the source environment to one database in the target environment. However, you can't map a database in the source environment to multiple databases in the target environment.

- If you migrate assets from a source database to a target database that doesn't exist and you have set up a database mapping, the mapping is used to set the database in the target environment so that you don't need to select it manually.
- If you migrate assets from a source database to a target database that exists and this contradicts the database mappings, the mapping is ignored and the assets are migrated as specified.
- If you migrate assets from a source database to a target database that doesn't exist and doesn't have a mapping set up, the database must be selected manually.

For more information, see "Correct errors and validate" on page 300.

Procedure

1. Click ⚙ on the Lifecycle Management page.
2. Click **New Mapping** to create a mapping.
3. Select a **Source Environment** from the drop-down list. The drop-down list is populated with all source environments that are available to you.
4. Select a **Target Environment** from the drop-down list. A database in the source environment is mapped to a database in the target environment.
5. Use the database selectors to change a database in the source environment or a database in the target environment.
6. Click **Add Mapping** to add another database to database mapping for this source and target environment.
7. Click ⊗ to remove a database mapping.
8. Continue to add or delete mappings until you are happy that you have mapped all the databases in the source environment to a database or databases in the target environment.
9. Click **Apply**.

Manage features

When some new features are introduced to Planning Analytics Workspace, you can choose to enable or disable the features in your environment. You can manage when and how your users are exposed to new functionality.

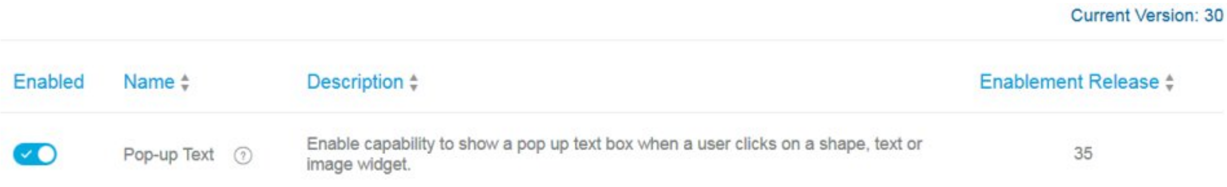
About this task

New features that are subject to management by administrators are disabled by default. You have the option of enabling some features when they are introduced in Planning Analytics Workspace. However, after the feature has been available for several release cycles, it becomes permanently enabled and is no longer subject to management by an administrator. The release in which the feature becomes permanent is referred to as the **Enablement Release** and it is clearly indicated for each feature.

Procedure

1. Click your user name on the Planning Analytics Workspace **Welcome** page.
2. Click **Administer**.
3. Click **Features**.

   The **Features** page lists all the features that you can manage and identifies the current version of Planning Analytics Workspace. Each feature is identified by its name, includes a brief description, and displays the **Enablement Release** for the feature.

   Current Version: 30

Enabled	Name	Description	Enablement Release
✓	Pop-up Text	Enable capability to show a pop up text box when a user clicks on a shape, text or image widget.	35

   When a feature is enabled, the button in front of the feature name displays a check mark - ✓.

   When a feature is disabled, the button in front of the feature name displays an 'x' - ✕.
4. To enable a feature that is currently disabled, click the button in front of the feature name, then click **Enable** when prompted for confirmation.
5. To disable a feature that is currently enabled, click the button in front of the feature name, then click **Disable** when prompted for confirmation.

Monitor and administer databases

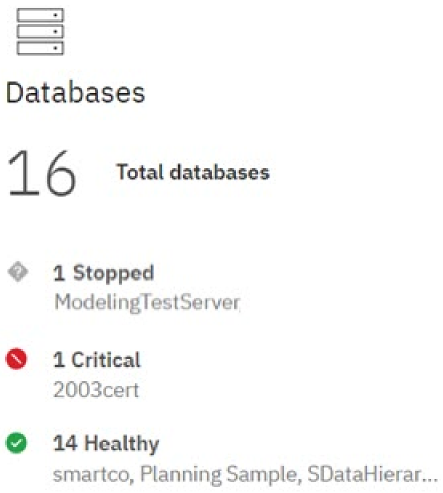
IBM Planning Analytics Administration includes the ability to monitor and administer your databases from the **Administration** page. You must be an administrator to access and use the **Administration** page.

Note: ⌂ To use Planning Analytics Administration on Planning Analytics Workspace Local, you must install and configure the Planning Analytics Administration agent wherever you have installed the Planning Analytics database. For more information, see "Install and configure the Planning Analytics Administration agent (local only)" on page 313.

To open the **Administration** page, click the **Administration** tile on the **Home** page.

Monitor and administer databases in Planning Analytics Workspace

The Databases tile on the **Administration** page shows the total number of databases in your environment, along with a status summary for all databases.



Databases

16 Total databases

1 Stopped
ModelingTestServer

1 Critical
2003cert

14 Healthy
smartco, Planning Sample, SDataHierar...

Databases are identified as being in one of these four states:

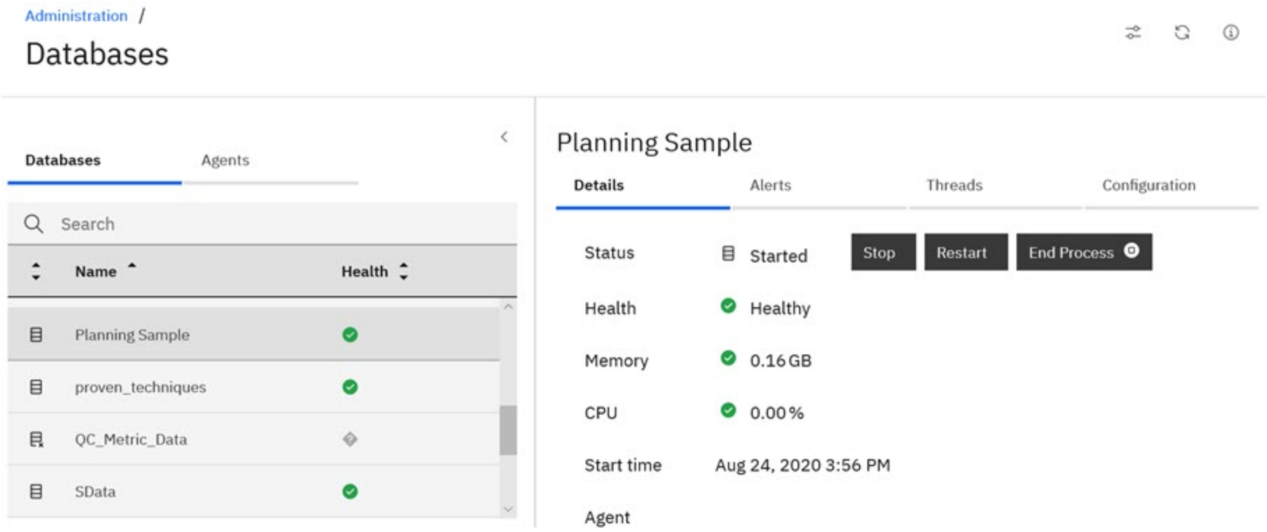
- ◈ Stopped - The database is nor running.
- 🚫 Critical - When any single database resource reaches the critical threshold, the entire database is in a critical state.
- ⚠ Concern - When any single database resource reaches the warning threshold, and there are no resources have reached the critical threshold, the entire database is in a warning state.
- ✓ Healthy - When all database resources are below the critical and warning thresholds, the entire database is in a healthy state.

Click the **Databases** tile to open the **Databases** page. This page shows a list of all the databases in your environment. For each database, you can see if it is running or stopped, the database name, and the database health.

Stopped databases are identified by the 🗄✕ icon, while running databases are identified by the 🗄 icon.

You can sort databases by running/stopped status, name, or health.

You can click any database name on the Databases list to view details for the database. The **Start time** reported for the database reflects the most recent database start or stop initiated directly through the Planning Analytics Workspace Administration page. If a database is started or stopped in any other way, such as through IBM Cognos Configuration or an API call, those times are not reported.



Administration /

Databases

	Name	Health
🗄	Planning Sample	✓
🗄	proven_techniques	✓
🗄	QC_Metric_Data	◈
🗄	SData	✓

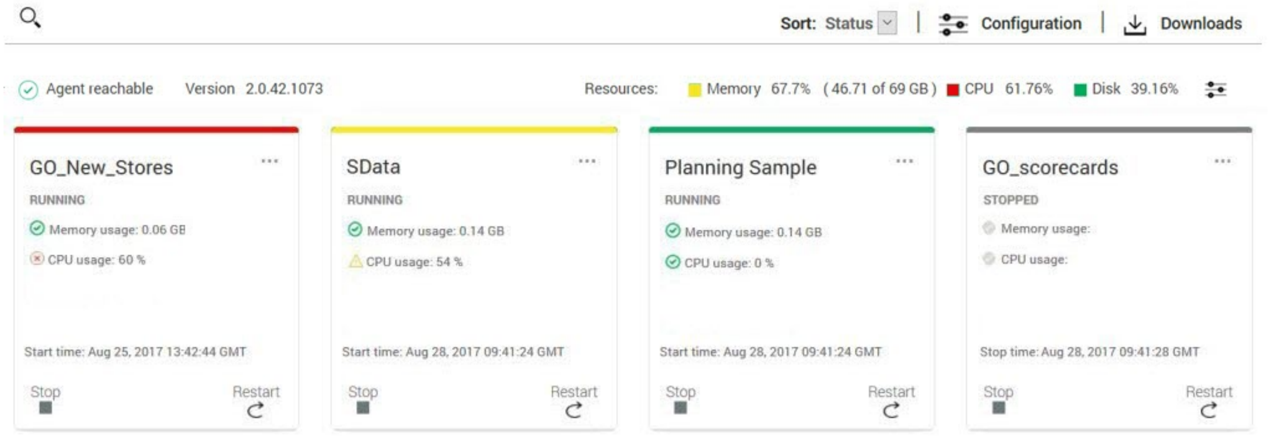
Planning Sample

Status	🗄 Started
Health	✓ Healthy
Memory	✓ 0.16 GB
CPU	✓ 0.00 %
Start time	Aug 24, 2020 3:56 PM
Agent	

In Planning Analytics Workspace Local, the Details page also indicates the agent on which the database is running.

Monitor and administer databases in Planning Analytics Workspace Classic

On the **Administration** page. click **Databases** to see the dashboard.



✓ Agent reachable Version 2.0.42.1073

Resources: 🟨 Memory 67.7% (46.71 of 69 GB) 🟥 CPU 61.76% 🟩 Disk 39.16%

GO_New_Stores
RUNNING
✓ Memory usage: 0.06 GB
⊗ CPU usage: 60 %
Start time: Aug 25, 2017 13:42:44 GMT

SData
RUNNING
✓ Memory usage: 0.14 GB
⚠ CPU usage: 54 %
Start time: Aug 28, 2017 09:41:24 GMT

Planning Sample
RUNNING
✓ Memory usage: 0.14 GB
✓ CPU usage: 0 %
Start time: Aug 28, 2017 09:41:24 GMT

GO_scorecards
STOPPED
Memory usage:
CPU usage:
Stop time: Aug 28, 2017 09:41:28 GMT

The dashboard shows the databases that are available on your system, along with performance metrics. Each database appears as an individual tile on the dashboard.

The **Start time** and **Stop time** reported for each database reflects the most recent database start or stop initiated directly through the Planning Analytics Monitoring dashboard. If a database is started or stopped in any other way, such as through IBM Cognos Configuration or an API call, those times are not reported.

For each database, you can view a combination of performance metrics. These metrics help you determine the efficiency and status of your databases. You can define the Safe, Concern, and Critical thresholds for all performance metrics. A quick glance at the dashboard gives you a visual indication of the health of your databases.

- When any metric meets or exceeds the Critical threshold value, the metric is identified with a red "x" icon ⊗. If any single metric is critical, the entire TM1 database is considered to be in a critical state, as indicated by a red bar at the top of the database tile.
- When any metric falls within the Concern threshold range, the metric is identified with a yellow "!" icon ⚠. If there are no metrics in a critical state, but at least one metric is in a concern state, the entire TM1 database is considered to be in a state of concern, as indicated by a yellow bar at the top of the database tile.
- When any metric is at or below the Safe threshold value, the metric is identified with a green check mark icon ✓. When all metrics are safe, the entire TM1 database is considered to be in a safe state, as indicated by a green bar at the top of the database tile.
- Any database that is stopped appears with a gray bar at the top of the database tile.
- For each database that is running, the most recent start time is displayed. For each stopped database, the most recent stop time is displayed.

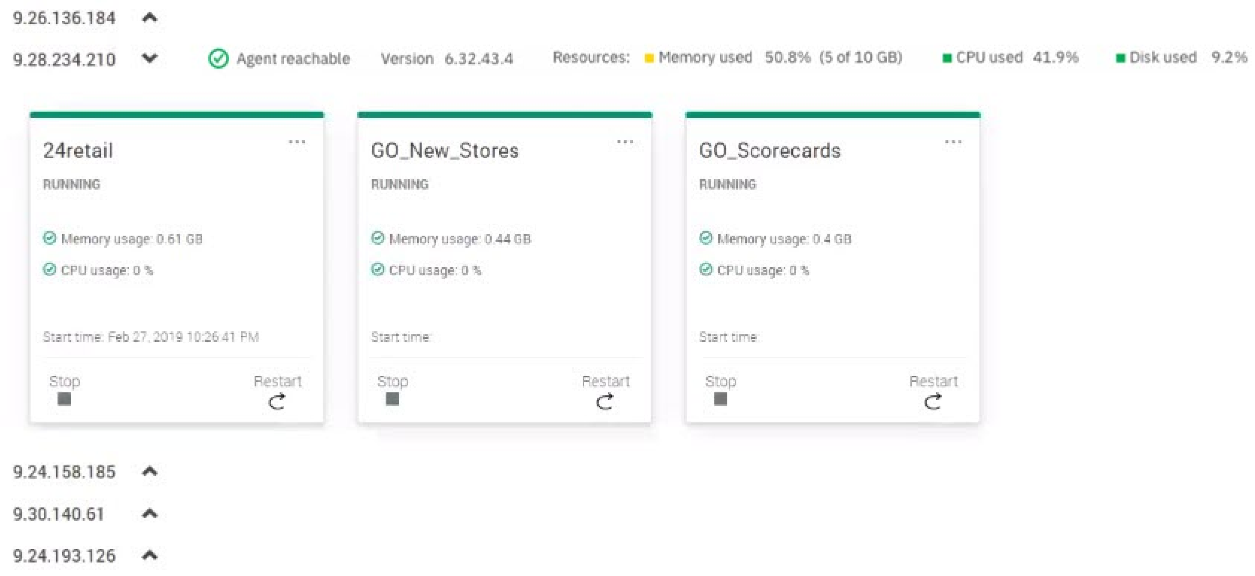
You can sort databases by status or name. When you sort by status, databases in a critical state appear first, followed by those in a concern state, followed by safe databases, with stopped databases appearing last. Sorting by name shows all databases in alphabetical order regardless of status.

You can click the database name on the tile of any running database to view a detailed report of the current database state.

In addition to showing the status for individual databases, the dashboard also shows the status of the total system resources on the computer where the databases are running. **Memory**, **CPU**, and **Disk** usage are displayed, showing the percentage of available resources currently used on the system. Each metric is accompanied by a color-coded square, indicating if resource usage is at the safe, concern, or critical level.

As with individual databases, you can define the Safe, Concern, and Critical thresholds for the system resources. When a system resource is in a safe state, the square for the resource is green. When in a concern state, the square is yellow. When in a critical state, the square is red.

For Planning Analytics Administration on Planning Analytics Workspace Local, databases are grouped by agent IP address. Click the arrow next to the agent address to reveal or hide the databases for that agent and to view the system resource status.



9.26.136.184

9.28.234.210 Agent reachable Version 6.32.43.4 Resources: Memory used 50.8% (5 of 10 GB) CPU used 41.9% Disk used 9.2%

24retail
RUNNING
Memory usage: 0.61 GB
CPU usage: 0 %
Start time: Feb 27, 2019 10:26:41 PM

GO_New_Stores
RUNNING
Memory usage: 0.44 GB
CPU usage: 0 %
Start time:

GO_Scorecards
RUNNING
Memory usage: 0.4 GB
CPU usage: 0 %
Start time:

9.24.158.185

9.30.140.61

9.24.193.126

What are the differences between Planning Analytics Administration on premises and on cloud?

The following database administration capabilities **are** available in Planning Analytics Workspace Local:

- “Monitor and manage database thread activity” on page 322
- “Start and stop databases” on page 331
- Download database log files
- Download the latest version of the Planning Analytics Administration agent

The following capabilities **are not** available in Planning Analytics Workspace Local:

- Secure Gateway
- Planning Analytics for Microsoft Excel download
- More than one Planning Analytics Administration agent
- System resources, thresholds, and server disk usage

Install and configure the Planning Analytics Administration agent (local only)

If you have IBM Planning Analytics Local version 2.0.5 or later installed, you can install and configure the Planning Analytics Administration agent for Microsoft Windows or Linux operating systems.

To use IBM Planning Analytics Administration on Planning Analytics Workspace Local, you must install and configure the Planning Analytics Administration agent wherever you have installed IBM TM1 Server. The default port of the Planning Analytics Administration agent is 9012.

By default, the Planning Analytics Administration agent is selected as a component when you install Planning Analytics Local but it is not configured or started.

Note: You can upgrade your Planning Analytics Administration agent by installing a new version of Planning Analytics Local or by downloading a new version of the agent from Planning Analytics Workspace. For more information, see Planning Analytics Administration agent in Download additional components.

To upgrade an Planning Analytics Administration agent as part of a Planning Analytics Local upgrade, you must stop the Planning Analytics Administration agent service, back up your `bootstrap.properties` file, upgrade Planning Analytics Local selecting the Planning Analytics Administration agent component in the installation wizard, and then restart the Planning Analytics Administration agent. For more information, see Upgrading Planning Analytics Local.

Your changes to the `bootstrap.propertes` file are preserved when you upgrade the Planning Analytics Administration agent, however, the version of the agent is updated. On Windows, the version is updated when you upgrade using the Planning Analytics Local installation wizard. On Linux, the version is updated when you run `./startup_agent.sh install`.

After you install or upgrade the agent, you must configure it. For more information, see "Configure the agent for Windows" on page 314 or "Configure the agent for Linux" on page 314.

Configure the agent for Windows

Procedure

1. Open the Windows Services desktop application.
2. Stop the **IBM Planning Analytics Administration Agent** service if it is running.
3. Navigate to *<PA_install_location>*`/paa_agent/wlp/usr/servers/kate-agent`.
4. In a text editor, open the `bootstrap.properties` file.
5. Set the SERVERS_DIR to the full path of the directory that contains TM1 databases directories.

   Note: Multiple paths must be separated by a semicolon. For example, SERVERS_DIR=`C:/tm1/samples/tm1/;C:/prod/servers/`.
6. Save and close the `bootstrap.properties` file.
7. Navigate to *<PA_install_location>*`/paa_agent/bin`.
8. Run `PAAAgentSetJavaHome.bat` to set JAVA_HOME for your Planning Analytics Administration agent.
9. Start the **IBM Planning Analytics Administration Agent** Windows service.

   Note: You can also navigate to *<PA_install_location>*`/paa_agent/bin` and run the `PAAAgentStart.bat` script to start the Planning Analytics Administration agent Windows service.

What to do next

Navigate to Planning Analytics Administration in Planning Analytics Workspace Local and verify that the TM1 databases appear.

Note: If you change the `bootstrap.properties` file later, you must restart the **IBM Planning Analytics Administration Agent** Windows service by running `PAAAgentStart.bat` script.

Configure the agent for Linux

Before you begin

Set the RunningInBackground parameter in your `tms1.cfg` file to `RunningInBackground=T`.

Note: The `RunningInBackground` parameter is required for Linux only. It is used to suppress the prompts displayed by TM1 Server. Since Planning Analytics Administration uses scripts to start and stop databases on TM1 Server, you must set `RunningInBackground=T` on Linux.

Procedure

1. Navigate to *<PA_install_location>*`/paa_agent/bin`.
2. Run the `./shutdown_agent.sh` command to stop the **IBM Planning Analytics Administration Agent** if it is running.
3. Navigate to *<PA_install_location>*`/paa_agent/wlp/usr/servers/kate-agent`.
4. In a text editor, open the `bootstrap.properties` file.
5. Set the full path of the directory that contains TM1 databases data directories to SERVERS_DIR.

Note: Multiple paths must be separated by a semicolon. For example, `/opt/ibm/cognos/tm1_64/samples/tm1/;/srv/prod/servers/`.

6. Save and close the `bootstrap.properties` file.
7. Navigate to `<`*`PA_install_location`*`>/paa_agent/bin`.
8. Run `./set_java_home.sh <Full path to JRE>` to set JAVA_HOME for your Planning Analytics Administration agent.
9. Run `./startup_agent.sh install` to set up the Planning Analytics Administration agent service.

   Note: You must have root or sudo privileges to perform this step.
10. Run `./startup_agent.sh` command to start the Planning Analytics Administration agent.

What to do next

Navigate to Planning Analytics Administration in Planning Analytics Workspace Local and verify that the TM1 databases appear.

Note: If you change the `bootstrap.properties` file later, you must restart the **IBM Planning Analytics Administration Agent** by running `./startup_agent.sh` command.

Configure event notifications

About this task

To get notifications from Planning Analytics Administration for Planning Analytics Local, you must configure the following functionality:

Procedure

1. Set the following bootstrap properties for SMTP notifications:

   For example, set these properties:

   ```
   SMTP_EMAIL_PORT=587
   SMTP_EMAIL_AUTH=true
   SMTP_EMAIL_HOST=example.com
   SMTP_EMAIL_USERNAME=user@example.com
   SMTP_EMAIL_PASSWORD=Analytics123
   PAA_EMAIL_ADDRESS=noreply@example.com
   ```

   The `PAA_EMAIL_ADDRESS` must be a registered alias, otherwise set it to a primary alias, for example, the user name `noreply@example.com`.
2. Optional: Set `SMTP_EMAIL_START_TLS_ENABLE=true` and add the certificates to the TLS certificate store:

   a) Run the following command in a command prompt on Windows or a terminal on Linux.

   ```
   openssl s_client -showcerts -starttls smtp -crlf -connect example.com:587
   ```

   This command prints two certificates that begin with "`-----BEGIN CERTIFICATE-----`" and end with "`-----END CERTIFICATE-----`".

   b) Copy these two certificates (including the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines) to `Example_1.cer` and `Example_2.cer` and remember the location.

   For example, `<PA install directory>/paa_agent/wlp/usr/servers/kate-agent/resources/security`. You use this location in the next step.

   c) Import `Example_1.cer` to the Planning Analytics Administration agent store by using the following command and the location of the `Example_1.cer` file from step "2.b" on page 315.

   On Windows:

   ```
   keytool.exe -importcert -keystore "<PA install directory>/paa_agent/wlp/usr/servers/kate-
   agent/resources/security/server_store.p12" -storetype PKCS12
   ```

```
-trustcacerts -storepassapplix -file "<PA install directory>/paa_agent/wlp/usr/servers/
kate-agent/resources/security/Example_1.cer" -alias example1
```

On Linux:

```
./keytool -import -keystore "<PA install directory>/paa_agent/wlp/usr/servers/kate-agent/
resources/security/server_store.p12" -storetype PKCS12
-trustcacerts -storepass applix -file "<PA install directory>/paa_agent/wlp/usr/servers/
kate-agent/resources/security/Example.cer" -alias example1
```

d) Repeat the previous step to import the second certificate, `Example_2.cer`.

What to do next

You can get notifications for chore failures, threads that are in a run state, threads in a wait state, and more. For more information, see Set system resource thresholds and alerts in the *Planning Analytics Workspace* documentation.

Sample bootstrap.properties file

```
# General properties
SERVERS_DIR=<PA_install_location>/samples/tm1
EXE_PATH=<PA_install_location>/bin64/tm1s.exe
SERVER_INFO_PATH=<PA_install_location>/paa_agent/paaAgentCache
TM1_SAMPLES_PATH=<PA_install_location>/samples/tm1

TM1_SERVER_INFO_PROCESS_TIME_INTERVAL=20000
NOTIFICATION_MEM_USAGE_INTERVAL=30000
NOTIFICATION_SERVER_UNRESPONSIVE_INTERVAL=40000
FILE_TAILER_REFRESH_INTERVAL=86400000
PAA_EMAIL_ADDRESS=administration@planning-analytics.ibmcloud.com
MODEL_UPLOAD=model_upload
SCRIPT_TIMEOUT=15

# SMTP Properties
SMTP_EMAIL_PORT=<port>
SMTP_EMAIL_AUTH=false
SMTP_EMAIL_HOST=<host>
SMTP_EMAIL_START_TLS_ENABLE=false
SMTP_EMAIL_USERNAME=<username>
SMTP_EMAIL_PASSWORD=<encrypted_pwd>

# Logging properties
com.ibm.ws.logging.console.log.level=OFF
com.ibm.ws.logging.trace.specification="*\=audit\:com.ibm.pa.kate.agent.*\=warning"
com.ibm.ws.logging.max.file.size=10
com.ibm.ws.logging.max.files=4

VERSION=1.0.nn.nnn
```

General properties

SERVERS_DIR

The full path of the directory that contains the TM1 databases directories.

Note: Multiple paths must be separated by a semicolon. For example, `SERVERS_DIR=C:/tm1/samples/tm1/;C:/prod/servers/`.

You can specify the parent directory of your databases. You don't need to specify each database separately.

EXE_PATH

Internal use only. Do not change.

SERVER_INFO_PATH

Internal use only. Do not change.

TM1_SAMPLES_PATH

Internal use only. Do not change.

TM1_SERVER_INFO_PROCESS_TIME_INTERVAL

Internal use only. Do not change.

Time in milliseconds between successive task executions. Used in the scheduler that checks system health.

NOTIFICATION_MEM_USAGE_INTERVAL
Internal use only. Do not change.

Not currently used.

NOTIFICATION_SERVER_UNRESPONSIVE_INTERVAL
Internal use only. Do not change.

Time in milliseconds between successive task executions. Used in the scheduler that checks for server unresponsiveness.

FILE_TAILER_REFRESH_INTERVAL
Internal use only. Do not change.

Time in milliseconds between successive task executions. Used in scheduler that tails server logs for every TM1 Server.

PAA_EMAIL_ADDRESS
Internal use only. Do not change.

MODEL_UPLOAD
Internal use only. Do not change.

SCRIPT_TIMEOUT
Internal use only. Do not change.

SMTP properties

SMTP_EMAIL_PORT
Set to a free port. For example, `2500`.

SMTP_EMAIL_AUTH
Default is `false`.

SMTP_EMAIL_HOST
Set based on your email server. For example, `localhost`.

SMTP_EMAIL_START_TLS_ENABLE
Default is `false`. Whether you need to configure certificates and set START_TLS_ENABLE=`true` depends on your email server configuration.

SMTP_EMAIL_USERNAME
Full email address. For example, example@my domain.com.

SMTP_EMAIL_PASSWORD
Email password for SMTP_EMAIL_USERNAME.

Logging properties

com.ibm.ws.logging.console.log.level
This setting controls the granularity of messages that go to the console.

When this log is enabled, a `console.log` is generated with the agent `message.log` at *<PA_install_location>*`/paa_agent/wlp/usr/servers/kate-agent/logs`.

Valid values are: INFO, AUDIT, WARNING, ERROR, and OFF.

Default is OFF.

com.ibm.ws.logging.trace.specification="*=audit:com.ibm.pa.kate.agent.*=warning"
This setting is used to selectively enable trace.

This entry refers to the agent `message.log`.

Valid values are: INFO, AUDIT, WARNING, ERROR, and OFF.

A restart of the Planning Analytics Administration agent is required for a logging value to take effect when you change this property.

com.ibm.ws.logging.max.file.size

The maximum size (in MB) that a log file can reach before it is rolled.

Default is 10. For more information, see Logging and Trace.

com.ibm.ws.logging.max.files

If an enforced maximum file size exists, this setting is used to determine how many of each of the logs files are kept.

Default is 4. For more information, see Logging and Trace.

Configure system monitoring

You can configure the interval at which database status is refreshed on the **Databases** page, as well as configure system resource thresholds and alerts.

Set the refresh interval for database status in Planning Analytics Workspace Classic

Set the refresh interval to determine how frequently the status for all databases is refreshed on the Databases page.

Procedure

1. Click ⚙ on the **Databases** page.
2. Set the **Auto refresh time** in seconds.
3. Click **Apply**.

Set agent resource thresholds and alerts

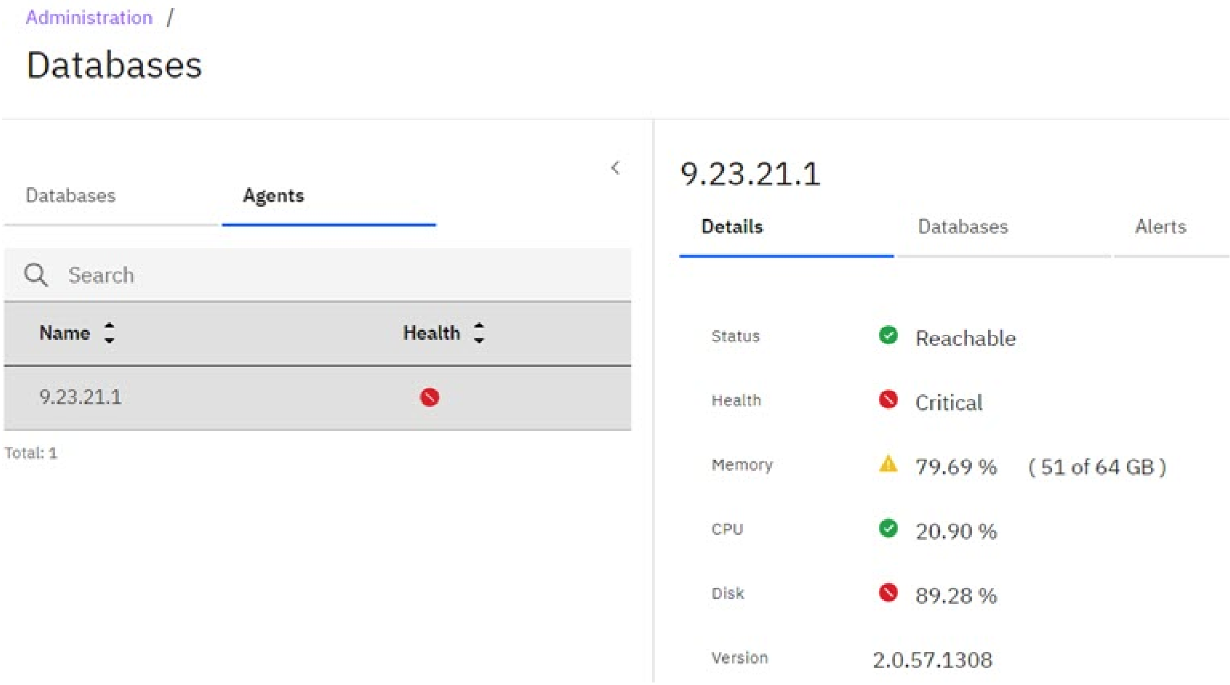
You can configure threshold settings that define agent resource status by multiple metrics. You can also configure alerts to send notification to yourself and other users when any of the system resources reaches a critical state.

About this task

When you set threshold limits, you define the parameters for the safe, concern, and critical states of these system resource metrics:

- Memory
- CPU
- Disk

In Planning Analytics Workspace, these states are displayed on the Databases page. A check mark in a green circle ✅ indicates a resource in a healthy state. A yellow triangle with an exclamation mark ⚠ indicates a resource in a concern state. A slash in a red circle ⛔ indicates a resource in a critical state.

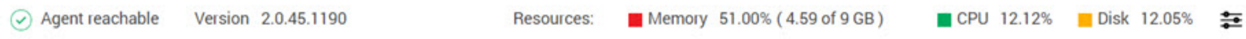


Name	Health
9.23.21.1	⛔

Total: 1

Status	✓ Reachable
Health	⛔ Critical
Memory	⚠ 79.69 % (51 of 64 GB)
CPU	✓ 20.90 %
Disk	⛔ 89.28 %
Version	2.0.57.1308

In Planning Analytics Workspace Classic, these states are displayed on the Databases page. A green square indicates a resource in a healthy state. A yellow square indicates a resource in a concern state. A red square indicates a resource in a critical state.



✓ Agent reachable Version 2.0.45.1190 Resources: 🟥 Memory 51.00% (4.59 of 9 GB) 🟩 CPU 12.12% 🟧 Disk 12.05%

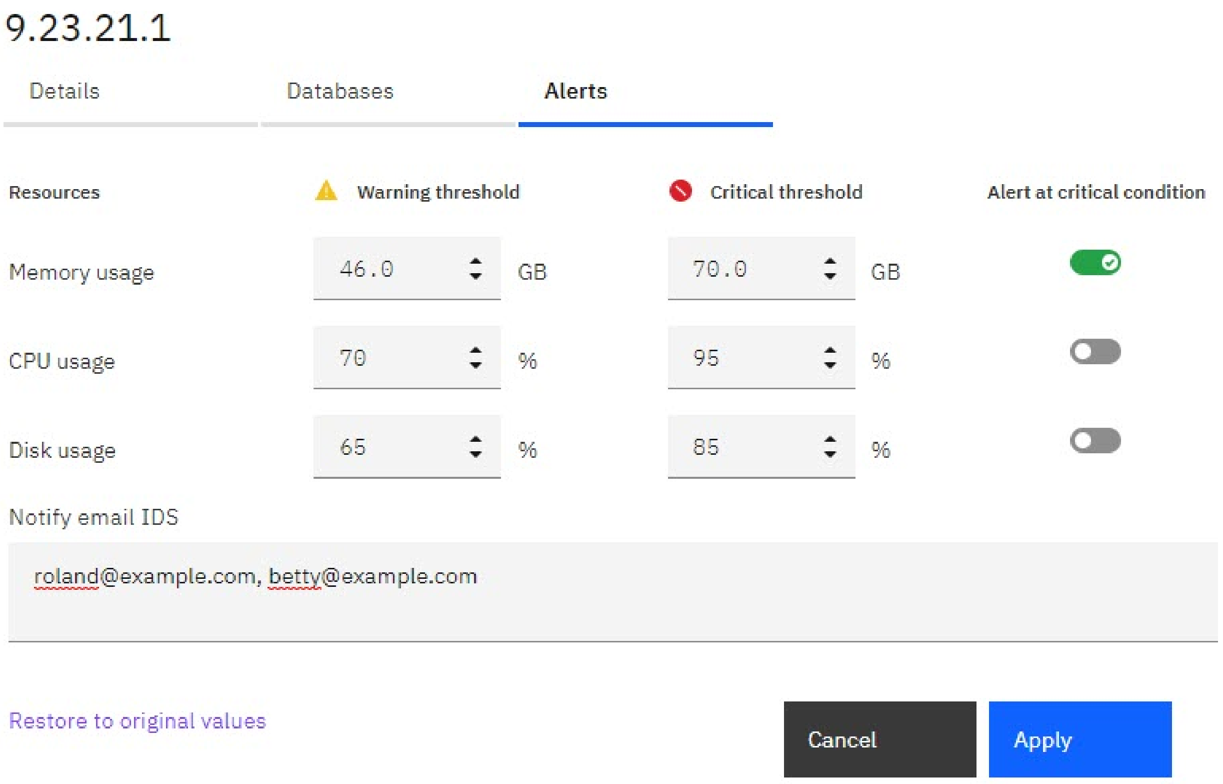
For Planning Analytics Administration on Cloud, threshold and alert settings apply to your entire environment.

⌂For Planning Analytics Administration Local, you must configure threshold and alert settings for each agent in your environment. Unique system resource metrics are displayed for each agent in your environment.

Set agent resource thresholds and alerts in Planning Analytics Workspace

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace Home page.
2. Click **Databases**.
3. Click the **Agent** tab on the **Databases** page.
4. Click the specific agent for which you want to set thresholds and alerts.
5. Click the **Alerts** tab.



9.23.21.1

Resources	Warning threshold		Critical threshold		Alert at critical condition
Memory usage	46.0	GB	70.0	GB	☑
CPU usage	70	%	95	%	☐
Disk usage	65	%	85	%	☐

Notify email IDS

roland@example.com, betty@example.com

6. For each system resource, set the **Warning threshold** and **Critical threshold**.

   When resource usage falls below the Warning threshold, the resource is in a healthy state. When resource usage is at or above the Warning threshold, but below the Critical threshold, the resource is in a concern state. When resource usage is at or above the Critical threshold, the resource is in a critical state.

   You can set thresholds for the following system resources:

   Memory usage

   Measures system memory usage in GBs.

   Default Warning threshold - 70 percent of installed system memory, expressed in GBs.

   Default Critical threshold - 80 percent of installed system memory, expressed in GBs.

   CPU usage

   Measures CPU usage as a percentage of total CPU capacity on the system.

   Default Warning threshold - 70

   Default Critical threshold - 95

   Disk usage

   Measures disk usage as a percentage of all system disk storage. The default **Concern at** range for Disk Usage is 65 to 85.

   Default Warning threshold - 65

   Default Critical threshold - 85

7. If you want to be alerted by email when a resource reaches a critical state:

   a) Select the **Alert at critical condition** option for the resource.

   b) Enter the email addresses of the users to receive notification in the **Notify email ID(s)** box. When typing, you can separate addresses with a comma, space, tab, or enter. You can also paste a comma-delimited list of email addresses in this box.

   For Planning Analytics Administration on Cloud, notification emails are sent from administration@planning-analytics.ibmcloud.com.

⌂For Planning Analytics Administration Local, notification emails are sent from the mail address specified by the PAA_EMAIL_ADDRESS property in the `bootstrap.properties` file for a given agent. The default value for this property is administration@planning-analytics.ibmcloud.com, but you can modify it to use an email address that is more relevant for your environment. See "Install and configure the Planning Analytics Administration agent (local only)" on page 313 for details on the `bootstrap.properties` file.

   c) Click **Apply**.

8. You can click **Restore to original values** at any time to restore the default threshold and alert settings. Note that **Restore** also removes any email addresses from the **Notify email ID(s)** box.

Set agent resource thresholds and alerts in Planning Analytics Workspace Classic

Procedure

1. Click ⚙ next to the system resources on the Databases page.
2. For each system resource, set the **Warning threshold** and **Critical threshold**.

   When resource usage falls below the Warning threshold, the resource is in a safe state. When resource usage is at or above the Warning threshold, but below the Critical threshold, the resource is in a concern state. When resource usage is at or above the Critical threshold, the resource is in a critical state.

   You can set thresholds for the following system resources:

   Memory usage

   Measures system memory usage in GBs.

   Default Warning threshold - 70 percent of installed system memory, expressed in GBs.

   Default Critical threshold - 80 percent of installed system memory, expressed in GBs.

   CPU usage

   Measures CPU usage as a percentage of total CPU capacity on the system.

   Default Warning threshold - 70

   Default Critical threshold - 95

   Disk usage

   Measures disk usage as a percentage of all system disk storage. The default **Concern at** range for Disk Usage is 65 to 85.

   Default Warning threshold - 65

   Default Critical threshold - 85

3. If you want to be alerted by email when a resource reaches a critical state:

   a) Select the **Alert at critical condition** option for the resource.

   b) Enter the email addresses of the users to receive notification in the **Notify email ID(s)** box. When typing, you can separate addresses with a comma, space, tab, or enter. You can also paste a comma-delimited list of email addresses in this box.

   For Planning Analytics Administration on Cloud, notification emails are sent from administration@planning-analytics.ibmcloud.com.

   ⌂For Planning Analytics Administration Local, notification emails are sent from the mail address specified by the PAA_EMAIL_ADDRESS property in the `bootstrap.properties` file for a given agent. The default value for this property is administration@planning-analytics.ibmcloud.com, but you can modify it to use an email address that is more relevant for your environment. See "Install and configure the Planning Analytics Administration agent (local only)" on page 313 for details on the `bootstrap.properties` file.

   c) Click **Apply**.

4. You can click **Restore** at any time to restore the default threshold and alert settings. Note that **Restore** also removes any email addresses from the **Notify email ID(s)** box.

Monitor and manage database thread activity

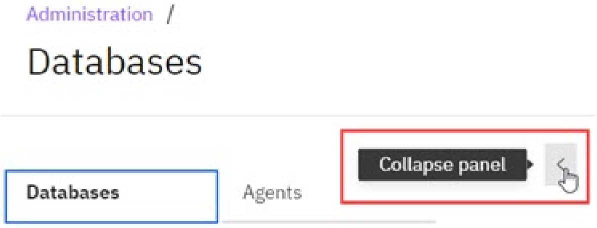
You can view a detailed report of thread activity on any running database. Using the information available in the report, you can cancel threads or disconnect users that are negatively affecting database performance.

Monitor and manage database thread activity in Planning Analytics Workspace

About this task

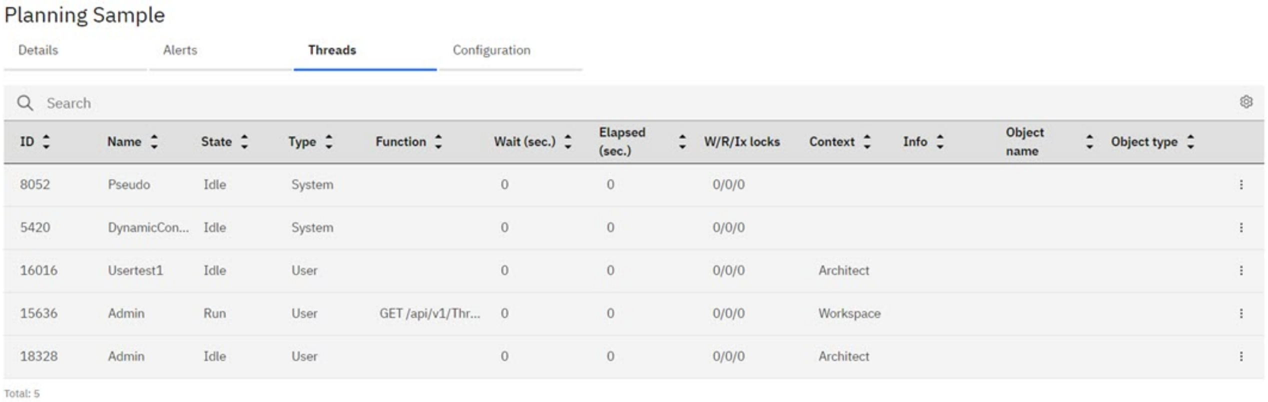
To open a database thread report, click the **Databases** tab on the Databases page, then click the database for which you want to view a report. Next, click the **Threads** tab to display the report.

Note: To view the full report, you might have to collapse the left pane on the **Databases** page.



When you open the **Databases** page as an administrator, you can see a list of all databases running in your environment. When you attempt to open a database thread report, Planning Analytics attempts to authenticate you as an administrator on the selected database using the same username and password you used to log into Planning Analytics. If your administrator credentials on the selected database are different from those you used to log into Planning Analytics, authentication will fail and you'll be prompted to provide a valid administrator username and password for the database.

A database thread report provides details on every thread on your database, in table format. You can't change the column widths in the report, but you can hover over any item in the report to see the full value.



ID	Name	State	Type	Function	Wait (sec.)	Elapsed (sec.)	W/R/Ix locks	Context	Info	Object name	Object type
8052	Pseudo	Idle	System		0	0	0/0/0				
5420	DynamicCon...	Idle	System		0	0	0/0/0				
16016	Usertest1	Idle	User		0	0	0/0/0	Architect			
15636	Admin	Run	User	GET /api/v1/Thr...	0	0	0/0/0	Workspace			
18328	Admin	Idle	User		0	0	0/0/0	Architect			

Total: 5

For each thread you can view these details:

ID
The unique numeric ID of the thread.

Name
The name of the user who initiated the thread.

State
The current thread state.

- Idle: not using resources
- Wait: waiting on other threads, blocked
- Running
- Commit: committing updates made by a thread

- Rollback: rolling back to a state before an update
- Login: when a user logs in and initiates a thread

Type
Indicates a User or System thread.

Function
The API function being executed by the thread.

Wait
The length of time, in seconds, that the thread has been in a wait state.

Elapsed
The total elapsed time of the thread in its current state.

W/R/Ix locks
The number of write, read, and intent-to-write locks that the thread holds.

A write lock allows only the thread that holds the lock to access and write changes to an object. No other thread can read or modify the object until the W lock has been released.

A read lock allows many threads to read from an object at the same time, but does not allow another thread to modify or write to the object until all R locks have been released.

An intent-to-write lock reserves the right for a thread to obtain a write lock on an object when all read locks have been released. Only one thread at a time is allowed to have an Ix lock on an object.

Context
The connected Planning Analytics client that is associated with the thread.

Info

When a thread is in a wait state, this column displays the ThreadID of the thread that is blocking the waiting thread.

Object name
The name of the object upon which the thread is operating.

Object type
The type of object upon which the thread is operating.

You can click ⚙ to selectively hide or show any of these columns in your report. The column display configuration is retained between sessions. The next time you open a thread report for any database, you'll see the same columns.

Procedure

1. To sort values by any column, click directly on a column name to cycle through the sort options.
2. To cancel a thread, click the **Thread actions** button ⋮, then click **Cancel thread**.
   You can cancel any user thread, including threads you own. You cannot cancel system threads.
3. To disconnect a user, click the **Thread actions** button ⋮, then click **Disconnect user**.
   You can disconnect user threads, but not system threads. You cannot disconnect yourself.

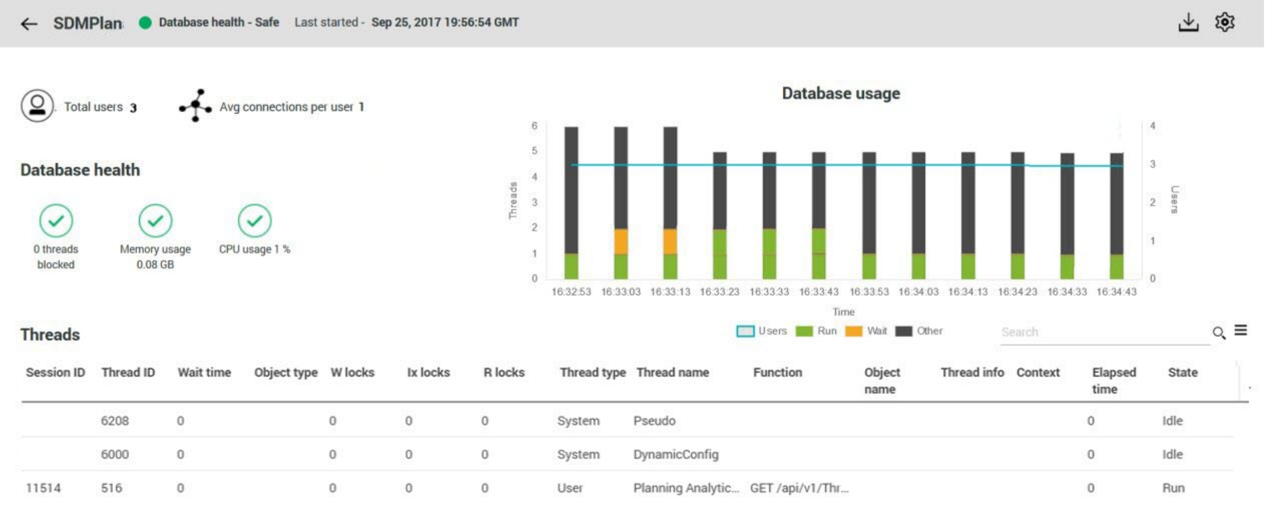
Monitor and manage database thread activity in Planning Analytics Workspace Classic

About this task

To open a database thread activity report, click the database name on the tile of any running database on the Planning Analytics Monitoring dashboard.

Note: When you open the **Databases** page as an administrator, you can see tiles for all databases running in your environment. When you click a tile name to open a database thread report, Planning Analytics attempts to authenticate you as an administrator on the database using the same username and password you used to log into Planning Analytics. If your administrator credentials on the database are different from those you used to log into Planning Analytics, authentication will fail and you'll be prompted to provide a valid administrator username and password for the database.

A database thread activity report provides a summary of current thread status, showing the number of threads in Run, Wait, and Other states. The report displays a chart of thread states over time and a table showing details on every thread on your database.



← SDMPlan ● Database health - Safe Last started - Sep 25, 2017 19:56:54 GMT

Total users 3 Avg connections per user 1

Database health

0 threads blocked

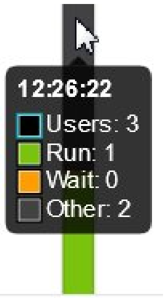
Memory usage 0.08 GB

CPU usage 1 %

Threads

Session ID	Thread ID	Wait time	Object type	W locks	Ix locks	R locks	Thread type	Thread name	Function	Object name	Thread info	Context	Elapsed time	State
	6208	0		0	0	0	System	Pseudo					0	Idle
	6000	0		0	0	0	System	DynamicConfig					0	Idle
11514	516	0		0	0	0	User	Planning Analytic...	GET /api/v1/Thr...				0	Run

You can hover over any bar on the **Database usage** chart to see details on users and thread states.



The report also shows the number of connected users and the average number of connections per user, along with database usage metrics. These are the same metrics that you configure on the Thresholds and alerts page.

You can click ⤓ to download database log files.

For each thread you can view these details, arranged as columns in the report.

Thread ID
The unique numeric ID of the thread.

State
The current thread state. One of the following states:

- Idle: not using resources
- Wait: waiting on other threads, blocked
- Running
- Commit: committing updates made by a thread
- Rollback: rolling back to a state before an update
- Login: when a user logs in and initiates a thread

Thread info

When a thread is in a wait state, this column displays the ThreadID of the thread that is blocking the waiting thread.

Wait time
The length of time, in seconds, that the thread has been in a wait state.

Thread type
Indicates a User or System thread.

W locks
The number of write locks the thread holds. A write lock allows only the thread that holds the lock to access and write changes to an object. No other thread can read or modify the object until the W lock has been released.

Function
The API function being executed by the thread.

Thread name
The name of the user who initiated the thread.

Object type
The type of object upon which the thread is operating.

Elapsed time
The total elapsed time of the thread in its current state.

Context
The connected client that is associated with the thread. For example, TM1 Architect, TM1 Perspectives, or Cognos Analytics.

Object name
The name of the object upon which the thread is operating.

R locks
The number of read locks the thread holds. A read lock allows many threads to read from an object at the same time, but does not allow another thread to modify or write to the object until all R locks have been released.

Ix locks
The number of intent-to-write locks the thread holds. An intent-to-write lock reserves the right for a thread to obtain a write lock on an object when all read locks have been released. Only one thread at a time is allowed to have an Ix lock on an object.

You can click ☰ to selectively hide or show any of these columns in your report.

Procedure

1. To sort values by any column, click directly on a column name to cycle through the sort options.

   When you apply sorting or hide columns, the column configuration is retained between sessions. The next time you open an activity report for any database, you'll see the same columns using the same sort order.
2. To cancel a thread, right-click any thread record, then click **Cancel thread**.

   You can cancel any user thread, including threads you own. You cannot cancel system threads.
3. To disconnect a user, right-click any thread record, then click **Disconnect user**.

   You can disconnect user threads, but not system threads. You cannot disconnect yourself.

Configure databases

You can set configuration parameters for your databases, set the thread auto-refresh interval for the database activity report, and configure system resource thresholds and alerts.

Set database configuration parameters

You can set configuration parameters that determine the behavior of your databases.

About this task

The Planning Analytics Administration interface provides access to many, but not all, Planning Analytics/TM1 database parameters. Only the parameters listed here can be set through Planning Analytics Administration.

Parameters are organized in **Administration**, **Modelling**, **Performance**, and **Access** categories, and then further arranged by functional category.

Each parameter includes a brief description, along with an indication of the default parameter value. For complete details on all database configuration parameters, see Parameters in the tm1s.cfg file.

Administration category

Default

AllowReadOnlyChoreReschedule

DisableSandboxing

Language

MaskUserNameInServerTools

PerformanceMonitorOn

RunningInBackground

UnicodeUpperLowerCase

External Database

SQLRowsetSize

TM1ConnectorforSAP

UseNewConnectorforSAP

TM1 Web

ExcelWebPublishEnabled

Modelling category

Startup

PersistentFeeders

SkipLoadingAliases

Rules

AllowSeparateNandCRules

AutomaticallyAddCubeDependencies

ForceReevaluationOfFeedersForFedCellsOnDataChange

RulesOverwriteCellsOnLoad

Spreading

SpreadingPrecision

Default

EnableNewHierarchyCreation

DefaultMeasuresDimension

UserDefinedCalculations

TI

EnableTIDebugging

UseExcelSerialDate

Performance category

MTQ (multi-threaded queries)

NumberOfThreadsToUse

SingleCellConsolidation

Memory

MaximumViewSizeMB

CacheFriendlyMalloc

ApplyMaximumViewSizeToEntireTransaction

DisableMemoryCache

LockPagesInMemory

Stargate

AllRuleCalcStargateOptimization

UseStargateForRules

ZeroWeightOptimization

Locking

UseLocalCopiesForPublicDynamicSubsets

View calculation

ViewConsolidationOptimization

Access category

CAM

CreateNewCAMClients

CAPI

MessageCompression

ProgressMessage

SSL

NIST_SP800_131A_MODE

Set database configuration parameters in Planning Analytics Workspace

Procedure

1. On the Databases page, click the **Databases** tab.
2. In the list of databases, click the database for which you want to set configuration parameters.
3. Click the **Configuration** tab.
4. Locate the parameter that you want to modify, then set a new parameter value.
5. Click **Apply**.

   Database configuration parameters are identified on the Configuration tab as either dynamic or static. Dynamic parameter values can be changed without requiring a database restart. Changes to static parameter values require a database restart.
6. You can click **Restore to original values** at any time to restore all of the listed parameters to their default values.

Set database configuration parameters in Planning Analytics Workspace Classic

Procedure

1. Click the database tile name on the Databases page.
2. Click ⚙ on the database activity report.
3. Click **Database configuration**.
4. Locate the parameter that you want to modify, then set a new parameter value.

5. Click **Apply**.

   Database configuration parameters are identified on the **Database configuration** page as either dynamic or static. Dynamic parameter values can be changed without requiring a database restart. Changes to static parameter values require a database restart.

   If you modify a static parameter, you receive notification that the database must be restarted before the new parameter value can be applied. You can restart the database from the Planning Analytics Monitoring dashboard.
6. You can click **Restore** at any time to restore all of the listed parameters to their default values.

Set the thread auto-refresh interval

By default, thread status on the database thread report is updated every 50 seconds. You can modify the refresh interval if you want to refresh thread status more frequently.

Set the thread auto-refresh interval Planning Analytics Workspace

Procedure

1. Click 🎚 on the Databases administration page.
2. Use the slider to set a different refresh interval

Set the thread auto-refresh interval In Planning Analytics Workspace Classic

Procedure

1. Click ⚙ on the database thread report.
2. Click **Thread auto-refresh**.
3. Use the slider to set a different refresh interval

Set database thresholds and alerts

You can configure settings that define database status by multiple metrics. You can also configure alerts to send notification to yourself and other users when any of the metrics reaches a critical state.

About this task

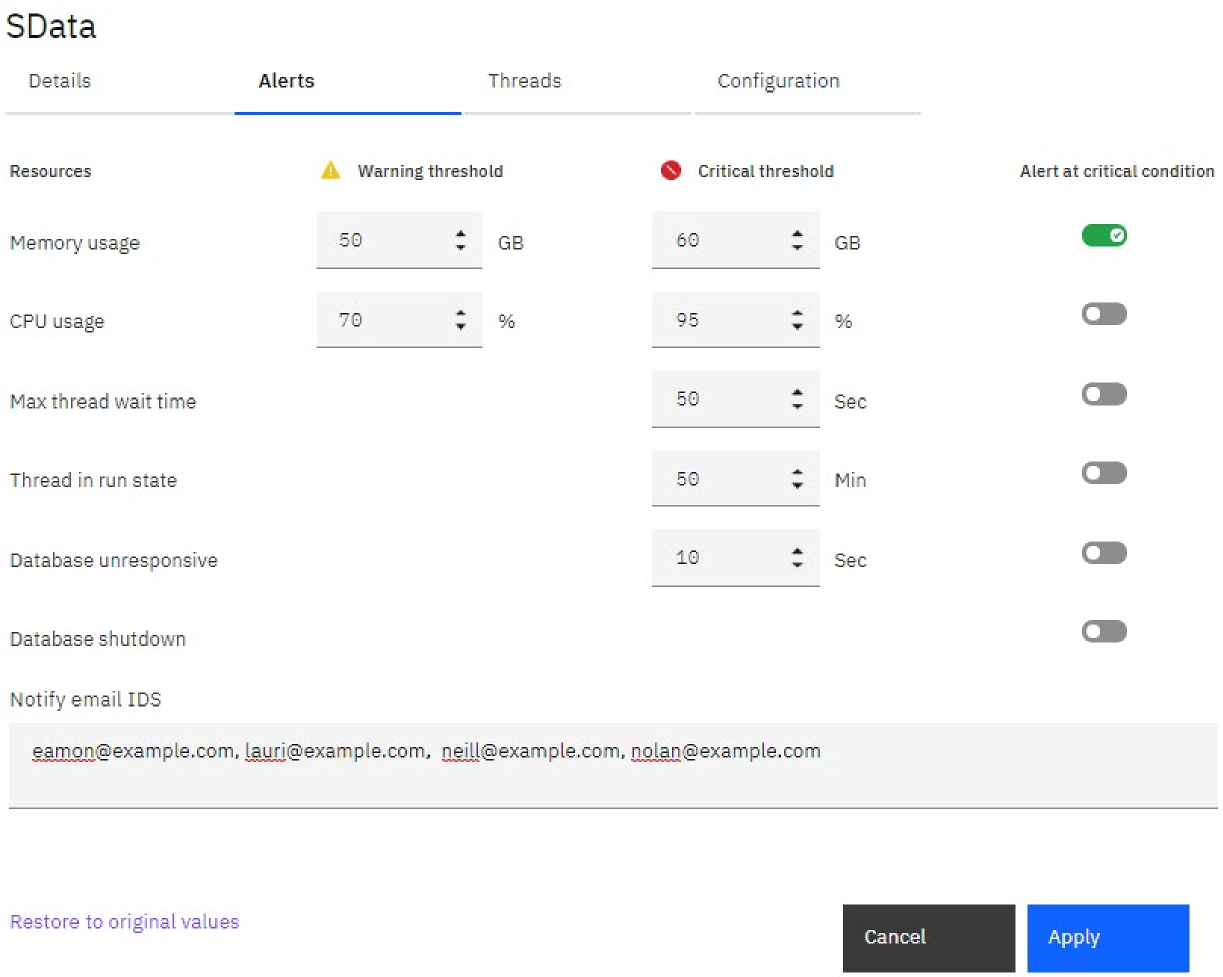
When you set threshold limits, you define the parameters for the safe, concern, and critical states of these metrics for an individual database:

- Memory usage
- CPU usage
- Maximum thread wait time
- Duration of thread in run state
- Database unresponsive

Set database thresholds and alerts in Planning Analytics Workspace

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace Home page.
2. Click **Databases**.
3. On the **Databases** tab, click the database for which you want to set thresholds and alerts.
4. Click **Alerts**.



Resources	⚠ Warning threshold	🛇 Critical threshold	Alert at critical condition
Memory usage	50 GB	60 GB	●
CPU usage	70 %	95 %	○
Max thread wait time		50 Sec	○
Thread in run state		50 Min	○
Database unresponsive		10 Sec	○
Database shutdown			○

Notify email IDS

eamon@example.com, lauri@example.com, neill@example.com, nolan@example.com

5. For the following database metrics, set the **Warning threshold** and **Critical threshold**.

   When the metric value falls below the Warning threshold, the database is in a healthy state. When the metric is at or above the Warning threshold, but below the Critical threshold, the database is in a concern state. When the metric is at or above the Critical threshold, the database is in a critical state.

   Memory usage
   Measures database memory usage in GBs.

   CPU usage
   Measures database CPU usage as a percentage of total CPU capacity on the system.

6. For the following database metrics, set the **Critical threshold**.

   When the metric value is at or above the Critical threshold, the database is in a critical state. Otherwise, the database is in a safe state.

   Max thread wait time
   Measures the amount of time a threads waits to run, expressed in seconds

   Thread in run state
   Measures the duration of a thread in a run state, expressed in minutes

   Database unresponsive
   Measures the duration of a database in an unresponsive state, expressed in seconds.

7. If you want to be alerted by email when a database metric reaches a critical state or when the database is shut down:

   a) Select the **Alert at critical condition** option for the metric or for database shutdown.

b) Enter the email addresses of the users to receive notification in the **Notify email ID(s)** box. When typing, you can separate addresses with a comma, space, tab, or enter. You can also paste a comma-delimited list of email addresses in this box.

   For Planning Analytics Administration on Cloud, notification emails are sent from administration@planning-analytics.ibmcloud.com.

   ⌂For Planning Analytics Administration Local, notification emails are sent from the mail address specified by the `PAA_EMAIL_ADDRESS` property in the `bootstrap.properties` file for a given agent. The default value for this property is administration@planning-analytics.ibmcloud.com, but you can modify it to use an email address that is more relevant for your environment. See "Install and configure the Planning Analytics Administration agent (local only)" on page 313 for details on the `bootstrap.properties` file.

c) Click **Apply**.

8. You can click **Restore to original values** at any time to restore the default threshold and alert settings. Note that **Restore** also removes any email addresses from the **Notify email ID(s)** box.

Set database thresholds and alerts in Planning Analytics Workspace Classic

Procedure

1. Click ⚙ on the database thread report.
2. Click **Thresholds and alerts**.
3. For the following database metrics, set the **Warning threshold** and **Critical threshold**.

   When the metric value falls below the Warning threshold, the database is in a safe state. When the metric is at or above the Warning threshold, but below the Critical threshold, the database is in a warning state. When the metric is at or above the Critical threshold, the database is in a critical state. The database state is reflected in the color bar at the top of each database tile on the Databases page.

   Memory usage
   Measures database memory usage in GBs.

   CPU usage
   Measures database CPU usage as a percentage of total CPU capacity on the system.

4. For the following database metrics, set the **Critical threshold**.

   When the metric value is at or above the Critical threshold, the database is in a critical state. Otherwise, the database is in a safe state.

   Max thread wait time
   Measures the amount of time a threads waits to run, expressed in seconds

   Thread in run state
   Measures the duration of a thread in a run state, expressed in minutes

   Database unresponsive
   Measures the duration of a database in an unresponsive state, expressed in seconds.

5. If you want to be alerted by email when a database metric reaches a critical state or when the database is shut down:

   a) Select the **Alert at critical condition** option for the metric or for database shutdown.

   b) Enter the email addresses of the users to receive notification in the **Notify email ID(s)** box. When typing, you can separate addresses with a comma, space, tab, or enter. You can also paste a comma-delimited list of email addresses in this box.

      For Planning Analytics Administration on Cloud, notification emails are sent from administration@planning-analytics.ibmcloud.com.

      ⌂For Planning Analytics Administration Local, notification emails are sent from the mail address specified by the `PAA_EMAIL_ADDRESS` property in the `bootstrap.properties` file for a given agent. The default value for this property is administration@planning-analytics.ibmcloud.com, but you can modify it to use an email address that is more relevant for your environment. See "Install and configure the Planning Analytics Administration agent (local only)" on page 313 for details on the `bootstrap.properties` file.

c) Click **Apply**.

6. You can click **Restore** at any time to restore the default threshold and alert settings. Note that **Restore** also removes any email addresses from the **Notify email ID(s)** box.

Start and stop databases

You can start, stop, or restart databases from the **Databases** administration page.

About this task

⌂ On Planning Analytics Workspace Local, you cannot use the Databases page to stop, start, or restart a database that is running as an application. The database must be running as a service if you want to manage the database from the Planning Analytics Monitoring dashboard.

Start and stop databases in Planning Analytics Workspace

Procedure

1. On the **Databases** administration page, click the **Databases** tab, then click the database you want to work with.
2. Click the **Details** tab for the database.
3. Click **Start** to start a database that is stopped.
4. Click **Stop** to stop a database that is running normally. Click **Apply** when you're asked for confirmation.
5. Click **Restart** to stop and then restart a database that is running normally. Click **Apply** when you're asked for confirmation.
6. If a database enters an unresponsive or deadlocked state, you can click **End Process** to end the database process. Click **Apply** when you're asked for confirmation.

   Note: Do not use this option to stop a database that is running normally. The **End Process** option should be reserved only for databases in an unresponsive state. The **End process** option kills all the running threads in the database. Any data that is not logged prior to ending the database process is lost.

Start and stop databases in Planning Analytics Workspace Classic

Procedure

1. On the appropriate database tile, click **Run** to start a database.
2. Click **Stop** to stop a database that is running normally. Click **Stop** again when you're asked for confirmation.
3. Click **Restart** to stop and then restart a database that is running normally. Click **Restart** again when you're asked for confirmation.
4. If a database enters an unresponsive or deadlocked state, you can click ⋯ on the unresponsive database tile. and then click **End Process** to end the database process. Click **End Process** when you're asked for confirmation.

   Note: Do not use this option to stop a database that is running normally. The **End Process** option should be reserved only for databases in an unresponsive state. The **End process** option kills all the running threads in the database. Any data that is not logged prior to ending the database process is lost.

Download database log files

You can download several types of logs for a database. See Planning Analytics and TM1 logs for descriptions of the log files.

Download database log files in Planning Analytics Workspace

Procedure

1. On the **Databases** page, click the database for which you want to download logs.
2. On the database Details tab, click **Download Logs**.
3. On the **Download Logs** page, select the log files you want to download.

   To make it easier to find the log files you need, you can click a column heading to sort by **Version** (file name), **Date**, or **Size**. You can also search for files.

   If a log file is locked by a process, selection of the locked file is disabled and the log file cannot be downloaded.

   Note: You can download up to 500 MB of log files. If the files you select to download total more than 500 MB, the **Download** button is disabled. You must clear selections until the total of files is less than 500 MB.

   Additionally, if the logging directory for your database contains more than 3000 files, Planning Analytics Workspace is unable retrieve the list of files and will throw an error. To resolve this issue, archive or delete old log files until there are less than 3000 files in the logging directory.
4. Click **Download Logs**.

   The log files are saved in a .zip archive using the database name, `<database_name>tm1serverLogs.zip` The .zip archive is saved in your `Downloads` directory.

Download database log files in Planning Analytics Workspace Classic

Procedure

1. On the **Databases** page, click ⋯ on a database tile.
2. Click **Download logs**.
3. On the **Download logs** page, select the log files you want to download.

   To make it easier to find the log files you need, you can click a column heading to sort by **All Versions** (file name), **Date**, or **Size**. You can also search for files.

   If a log file is locked by a process, selection of the locked file is disabled and the log file cannot be downloaded.

   Note: You can download up to 500 MB of log files. If the files you select to download total more than 500 MB, the **Download** button is disabled. You must clear selections until the total of files is less than 500 MB.

   Additionally, if the logging directory for your database contains more than 3000 files, Planning Analytics Monitoring is unable retrieve the list of files and will throw an error. To resolve this issue, archive or delete old log files until there are less than 3000 files in the logging directory.
4. Click **Download**, then choose a location to save the log files.

   The log files are saved in a .zip archive using your database name, `<database_name>tm1serverLogs.zip`.

   You can also download log files from the database activity report page. Click ⤓.

Download additional components

You can download the following components of IBM Planning Analytics directly from Planning Analytics Administration.

Planning Analytics for Microsoft Excel (cloud only)

☁ To download Planning Analytics for Microsoft Excel, follow these steps.

Download Planning Analytics for Microsoft Excel in Planning Analytics Workspace

Procedure

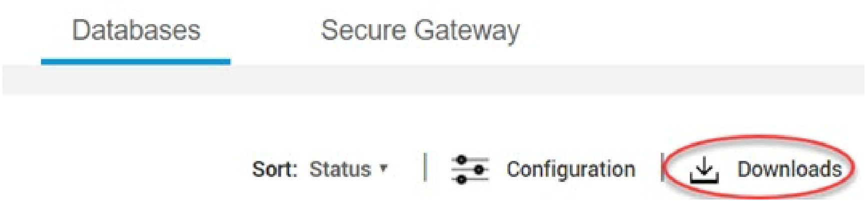
1. On the Planning Analytics Workspace Home page, click the **Administration** tile.
2. On the Administration page, click **Excel and Customizations**.
3. Select the version of Planning Analytics for Microsoft Excel you want to download. The three most recent versions are available for selection.
4. Click **Download Update**.

   The installation is saved as an archive named Integration<version>.zip in your Downloads directory.
5. Extract the archive.
6. Consult the `README.txt` file in the extracted archive for installation instructions.
7. Optional: Click the links to the **Installation guide** and **New features** to learn more about Planning Analytics for Microsoft Excel.

Download Planning Analytics for Microsoft Excel in Planning Analytics Workspace Classic

Procedure

1. Navigate to the Planning Analytics Administration dashboard.



2. Click **Downloads**. The Downloads dialog box opens.
3. To download Planning Analytics for Microsoft Excel, click the **IBM Planning Analytics for Microsoft Excel** tab.
4. Select the version of Planning Analytics for Microsoft Excel you want to download.
5. When prompted, save the download archive to the location of your choice.
6. Extract the archive.
7. Consult the `README.txt` file in the extracted archive for installation instructions.
8. Optional: Click the links to the **Installation guide** and **New features** to learn more about Planning Analytics for Microsoft Excel.

Planning Analytics Administration agent (local only)

⌂ To upgrade the Planning Analytics Administration agent, follow these steps.

Before you begin

- You must have IBM Planning Analytics Local version 2.0.5 or later installed wherever you have installed IBM TM1 Server.
- You must ensure that port 9012 is available. The default port of the Planning Analytics Administration agent is 9012.

- You must install and configure the Planning Analytics Administration agent. For more information, see "Install and configure the Planning Analytics Administration agent (local only)" on page 313.

Download the Planning Analytics Administration agent in Planning Analytics Workspace

Procedure

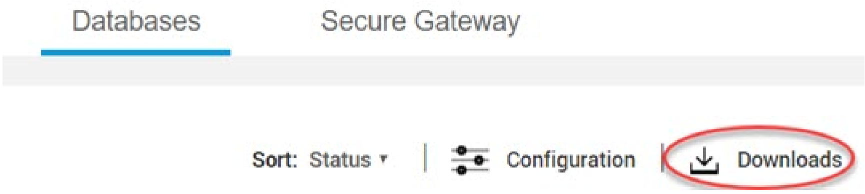
1. On the Planning Analytics Workspace Home page, click the **Administration** tile.
2. On the Administration page, click **Agents**.
3. Click the **Download** button ⤓, then click **Download Agent**.

   The installation is saved as an archive named `paa-agent-pkg-<version>.zip` in your Downloads directory.

Download the Planning Analytics Administration agent Planning Analytics Workspace Classic

Procedure

1. Navigate to the Planning Analytics Administration page.



2. Click **Downloads**. The Downloads dialog box opens.
3. To download the Planning Analytics Administration agent, click the **PAA Agent** tab.
4. Select the version of Planning Analytics Administration agent that you want to download.

   Only the latest version of the Planning Analytics Administration agent is available for download. You cannot downgrade your version of the Planning Analytics Administration agent from this dialog box.
5. When prompted, save the archive to the location of your choice.
   For example, save `paa-agent-pkg-1.0.<sc>.<nnn>.zip`. The *<sc>* is the release of Planning Analytics Workspace and *<nnn>* is the version of the Planning Analytics Administration agent.

Update the Planning Analytics Administration agent

Procedure

1. Extract the archive to a temporary location. .
2. Open a command window and change to the directory where you extracted the compressed folder.

   Note: You will not be able to stop a service or modify files in *`<PA_install_location>`*`\paa_agent\` unless you have the required permissions. On Windows, open the command window with `Run as administrator`.

   Tip: Put the full path to *`<PA_install_location>`* in quotation marks if it contains spaces.

   a) To update the Planning Analytics Administration agent Windows service, run **UpdatePAAAgent.bat**, passing in the full path to *`<PA_install_location>`*, that is, the parent of the `paa_agent` directory.

   b) To update the Planning Analytics Administration agent service on Linux, run **UpdatePAAAgent.sh**, passing in the full path to *`<PA_install_location>`*, that is, the parent of the `paa_agent` directory.

   The **UpdatePAAAgent** script stops the Planning Analytics Administration agent before updating it.

   Your existing `bootstrap.properties` file is preserved and updated with the new agent version number. For example, `VERSION=1.0.39.736`.

What to do next

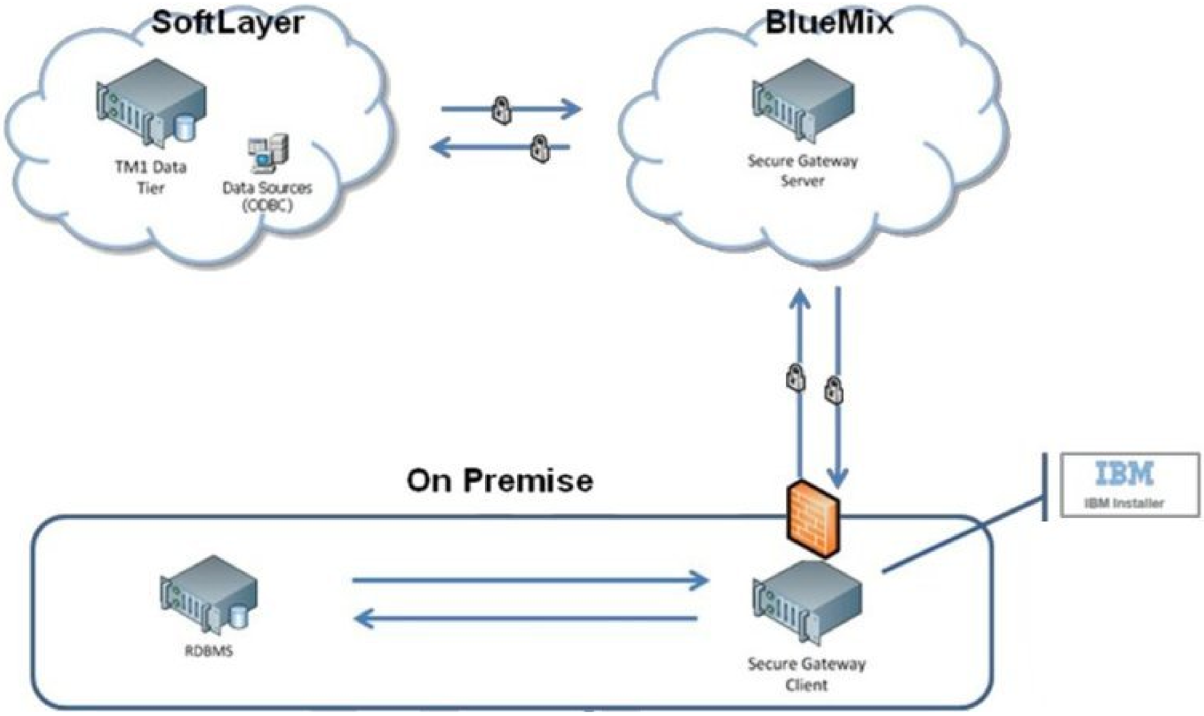
Note: The **UpdatePAAAgent** script restarts the Planning Analytics Administration agent at the end of the script.

Administer IBM Secure Gateway (cloud only)

IBM Secure Gateway creates a secure connection and establishes a tunnel between IBM Planning Analytics TM1 on cloud and an on - premise data source, typically an RDBMS source.

You can use IBM Secure Gateway to create and manage a secure connection between your on-cloud Planning Analytics environment and your on-premise data sources. Planning Analytics commonly contains source data representing summarized transaction data from ERP systems. These source systems are typically relational and accessed via ODBC using TurboIntegrator. The Secure Gateway allows your Planning Analytics components to interact seamlessly and securely with your on-premises data sources.

You must create a Secure Gateway if you want to access RDBMS data sources on-premises using TurboIntegrator. This is useful for importing data into TM1 and for drilling through to transactional data.



For complete details on IBM Secure Gateway, see Secure Gateway overview.

Create a Secure Gateway

Create an IBM Secure Gateway to establish a secure connection between your on-cloud Planning Analytics environment and your on-premises data sources. You can create up to 50 Secure Gateways on your system.

Create a Secure Gateway in Planning Analytics Workspace

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace **Home** page.
2. On the Administration page, click the **Secure Gateway** tile.
3. Click ⊕ at the top of the Secure Gateway list.

4. Enter a **Name** for your Secure Gateway.
5. Optionally, enter a **Description** for the gateway.
6. Set a **Token Expiration** period of between 90 and 365 days. The default expiration is 90 days.

   The security token is required. It authorizes access to the IBM Secure Gateway.
7. Click **Continue**.

Results

Your new gateway is created and enabled. The gateway name is added to the Secure Gateway list, which displays all the gateways available on your system. The new Secure Gateway **Details** tab is open and shows the details for your new gateway.

What to do next

You can now "Add a data source to a Secure Gateway" on page 337.

Create a Secure Gateway in Planning Analytics Workspace Classic

Procedure

1. Click the Administration tile on the Planning Analytics Workspace Welcome page.

   The Planning Analytics Administration page opens in a new web browser tab.
2. Click the **Secure Gateway** tab.
3. Click ⊕ at the top of the Secure Gateway list.
4. Enter a **Name** for your Secure Gateway.
5. Optionally, enter a **Description** for the gateway.
6. Set a **Token Expiration** period of between 90 and 365 days. The default expiration is 90 days.

   The security token is required. It authorizes access to the IBM Secure Gateway.
7. Click **Continue**.

Results

Your new gateway is created and enabled. The gateway name is added to the Secure Gateway list, which displays all the gateways available on your system.

What to do next

You can now "Add a data source to a Secure Gateway" on page 337.

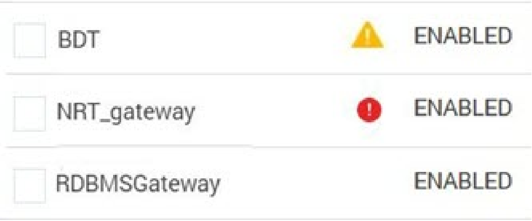
Renew a Secure Gateway security token

You can renew a Secure Gateway security token at any time, even if the token has expired.

About this task

If a security token will expire within fourteen days, a yellow alert triangle appears next to the gateway name.

If a security token has already expired, a red alert circle appears next to the gateway name.



□	BDT	⚠	ENABLED
□	NRT_gateway	●	ENABLED
□	RDBMSGateway		ENABLED

Additionally, security token status is displayed on the Details tab when a token is expired or soon to expire.

Procedure

1. Next to the security token on the Details tab, enter a new token expiration period of up to 365 days.
2. Click ↻.

Delete a Secure Gateway

Procedure

1. On the **Secure Gateway** tab, select the Secure Gateway(s) that you want to delete. You can click the box next to the **Name** heading to select all gateways.
2. Click 🗑 at the top of the **Secure Gateway** list.

   If any of the selected Secure Gateways has an expired security token, the Delete icon is disabled. You must deselect any Secure Gateways that have expired security tokens before you can proceed with deletion.
3. Click **Delete** when prompted for confirmation.

Add a data source to a Secure Gateway

After you create a Secure Gateway, you must establish a secure connection between your on-cloud Planning Analytics environment and a specific on-premises data source. You can add multiple data sources to a single Secure Gateway.

About this task

You can review a report listing all the databases supported by Secure Gateway at https://www.ibm.com/software/reports/compatibility/clarity-reports/report/html/prereqsForProduct?deliverableId=1413345793248.

Note that any Microsoft SQL Server data source must be patched per the following Microsoft Support guide: https://support.microsoft.com/en-us/help/3135244/tls-1-2-support-for-microsoft-sql-server. If this patch is not applied, you will receive an "Unable to open connection to database" error when establishing a connection to the SQL Server data source.

Please contact IBM support if you want to use Secure Gateway to establish a connection to a database that uses ODBC 3.5 drivers, but is not included in the list of supported databases.

Procedure

1. On the **Secure Gateway** page, click the gateway to which you want to add a data source from the list of available gateways.
2. On the Secure Gateway pane, click the **Data sources** tab.

   The Data sources tab displays all the data sources that have been defined for your gateway.
3. Click ⊕ at the top of the Data sources tab.
4. Enter the data source name, a description, and the data source host IP address:port.
5. Select a protocol for the connection, then click **Create**.
6. Select the appropriate driver from the **Driver** list, enter the database name, and then click **Create DSN**.
7. To test the DSN connection, enter your username and password, then click **Test DSN**. If the test is successful, your data source connection is complete and you can start using your Secure Gateway.

When you initially create a new data source, it is enabled by default. You can selectively enable or disable a data source at any time. Click the ellipses icon ⋯ for the data source on the Data Sources tab, then click **Enable** or **Disable**.

After a data source is created, you can click the data source name on the Data sources tab to modify data source parameters.

Configure Microsoft Access as a Secure Gateway data source

Some unique steps are required to configure Microsoft Access as a Secure Gateway data source.

Procedure

1. Use FTPS to copy all Access database files to a folder under `/prod` (or `/prod/data` for some systems). For example, `/prod/AccessDB` or `/prod/data/AccessDB`.
2. In the list of available gateways on the **Secure Gateway** page, click the gateway to which you want to add a Microsoft Access data source.
3. On the Secure Gateway pane, click the **Data sources** tab.

   The Data sources tab displays all the data sources that have been defined for the selected gateway.
4. Click ⊕ at the top of the Data sources tab.
5. Enter the data source name, an optional description, and the data source host IP address:port.
6. Select TCP as the protocol, then click **Create**.
7. Select Microsoft Access Driver or Microsoft Excel Driver from the **Driver** menu.
8. For **Database Name**, enter the path to the Access database on the Shared Data Drive starting with `S:\ `. For example, `S:\prod\AccessDB\my_access_database.accdb`.
9. Click **Create DSN**.
10. Click **Done**.

Delete a Secure Gateway data source

Procedure

1. On the Secure Gateway page, click the Secure Gateway that contains the data source(s) you want to delete.
2. Click the **Data sources** tab.
3. Select the data sources you want to delete. You can click the box next to the Name heading to select all data sources.
4. Click 🗑 on the **Data sources** tab.
5. Click **Delete** when prompted for confirmation.

⚠ **Warning:** When you delete a Secure Gateway data source in Planning Analytics Administration, the data source name is removed from the list of available data sources on the Secure Gateway. However, the data source is **not** deleted as an ODBC data source on the Planning Analytics data tier. This can cause difficulty in the following scenarios:

- If you try to create a new data source with the same name as the removed data source, you will receive an error indicating that the name is already in use.
- When creating a drill process to support drill through to an ODBC source, the data source will appear as available, because it has not been deleted from the data tier. However, you will not be able to establish a drill through connection because the data source is no longer configured and available on Secure Gateway.

This issue will be resolved in a future release.

Install the Secure Gateway client

The Secure Gateway client must be installed in a location that has access to the on-premise data source. The client uses the ODBC driver on the TM1 data tier to perform the actual ODBC connection between the on - premise data source and the TM1 data tier.

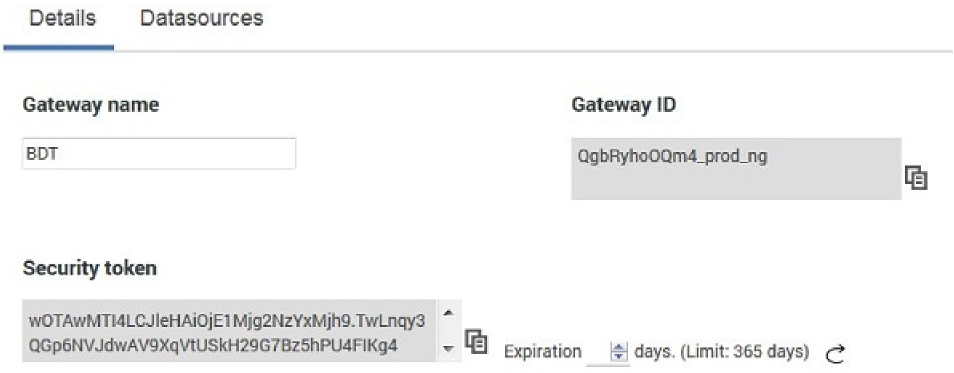
Before you begin

You can review the system, network, and hardware requirements for Secure Gateway at https://cloud.ibm.com/docs/services/SecureGateway?topic=securegateway-client-requirements#requirements-to-run-the-client.

Procedure

1. On the Secure Gateway page in Planning Analytics Administration, click ⤓ at the top of the list of available gateways.
2. Select the appropriate operating system for your Secure Gateway client, then click **Download**.
3. Save the client installer to a location of your choice.
4. Run the installer.

   While installing the Secure Gateway client you will be asked to provide the gateway ID and security token of the Secure Gateways you have installed and configured. You can find this information on the Details tab for each Secure Gateway. You can click ⧉ to copy the gateway ID or security token to the clipboard.

   Gateway name: BDT

   Gateway ID: QgbRyhoOQm4_prod_ng

   Security token: wOTAwMTI4LCJleHAiOjE1Mjg2NzYxMjh9.TwLnqy3QGp6NVJdwAV9XqVtUSkH29G7Bz5hPU4FIKg4

   Expiration days. (Limit: 365 days)

   For further details on installing the client on each supported operating system, see https://cloud.ibm.com/docs/services/SecureGateway?topic=securegateway-client-install#installing-the-client.

Use the Secure Gateway client

You can use the Secure Gateway client to maintain and upload access control lists, view and download logs, and monitor gateway activity.

There are several ways to access and use the Secure Gateway client. You can access the client:

- through terminal command line prior to startup.
- after startup, using the client's interactive command line.
- after startup, with the client's local UI.

For complete details on using the Secure Gateway client, see https://cloud.ibm.com/docs/services/SecureGateway?topic=securegateway-client-interacting#interacting-with-the-client.

Customize Planning Analytics Workspace

Administrators can upload fonts and color palettes to be used in IBM Planning Analytics Workspace.

Upload custom fonts

Administrators can upload custom fonts that can be used in views and text boxes.

You can upload the following file types: .tff (True Type), .otf (OpenType), .woff (Web Open Font Format), .eot (Embedded OpenType), .svg (Scalable Vector Graphics), and .woff2 (Web Open Font 2.0 Format).

Note: Not all font types are supported by all browsers. The font type works only if it is supported by the browser.

The font file naming convention is:

`<family name>-<style identifiers>.<extension>`

For example:

- `IBMPlexSans-Thin.ttf`
- `IBMPlexSans-BoldItalic.ttf`

The formatting enables font families to be grouped in the text properties.

Upload custom fonts

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace Home page.
2. Click the **Excel and Customizations** tile.
3. Click the **Fonts** tab.
4. Click the **Upload** ⤒ icon.
5. Either drag the font file onto the **Drag your font file here** box, or click **Tap to browse**, and select the font file.
6. Close any opened books and then reopen them.

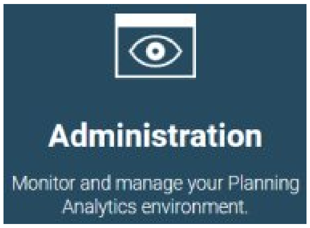
Results

The font is now available for selection in **Text Properties**.

Upload custom fonts

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace Welcome page.



   The Planning Analytics Administration page opens in a new web browser tab.
2. Click the **Settings** tab.
3. Click **Add custom fonts**.
4. Either drag the font file onto the **Drag your font file here** box, or click **Tap to browse**, and select the font file.

5. Close any opened books and then reopen them.

Results

The font is now available for selection in **Text Properties**.

Add or remove color palettes for charts in Planning Analytics Workspace Classic

Administrators in Planning Analytics Classic can change the color palettes that users can apply to charts. You can add new palettes, and edit or remove existing palettes.

Palettes are stored in themes, in JSON format.

About this task

To change the color palettes that are available, edit the theme that the color palette applies to.

Procedure

1. Click the **Administration** tile on the Planning Analytics Workspace Welcome page.

   Administration

   Monitor and manage your Planning Analytics environment.

   The Planning Analytics Administration page opens in a new web browser tab.
2. Click the **Settings** tab.
3. To add, change or remove a color palette, follow these steps:
   a) Click **Export** for the theme that is used for your plans.

      A JSON file is saved to your computer: `defaultTheme.json`, `lightTheme.json`, or `dark.Theme.json`.

   b) Open the theme in a suitable editor and search for `ColorPalette`.

      There are eight color palettes, which are identified by `"id": "colorPalette0"`, `"id": "colorPalette1"`, "id": "colorPalette2", and so on.

   c) You can add new palettes, delete existing palettes, and modify existing palettes by editing the file.

      This example shows the correct format for a color palette. If there are any errors in the file, the file won't load.

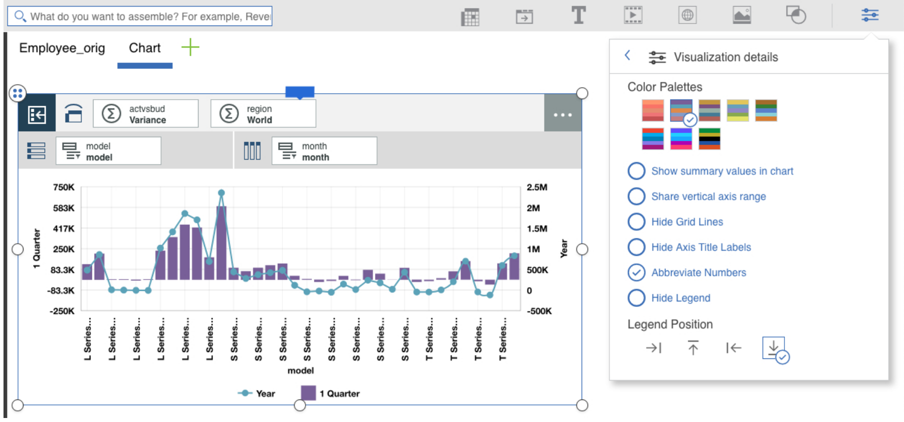
   d) Click **Upload theme override** next to the theme, and drag the updated JSON file onto the pane.
4. To reset the theme back to the original setting, click **Remove theme override**.

   Resetting a theme removes all updates.

   Note: If you want to make further changes to a theme, you must remove the previous theme override before you upload a new theme override.
5. Test the palette changes.
   a) Open a book and create a visualization, see “Visualizations in Planning Analytics Workspace Classic” on page 81.

   b) Select the visualization and click **Properties** > **Visualization details**.

   c) Select the new color palette, then click to close the menu.



Visualization details

Color Palettes

○ Show summary values in chart

○ Share vertical axis range

○ Hide Grid Lines

○ Hide Axis Title Labels

✓ Abbreviate Numbers

○ Hide Legend

Legend Position

Note: To see the changes in an open book, you must close and reopen the book.

Color palette

The color values are in hexadecimal format.

```
{
        "id": "colorPalette1",
        "caption": "Color Palette",
        "value": [
          "#836aa1",
          "#66abc0",
          "#e29557",
          "#91a6ce",
          "#ca898c",
          "#b6cc8d",
          "#a594bb",
          "#91c3d4",
          "#f2b087",
          "#3b86b9",
          "#b4595c",
          "#99b65d"
        ],
        "heatMapping": [
          {
            "at": "0.0%",
            "color": "#F2E6FF"
          },
          {
            "at": "25.0%",
            "color": "#B49FCC"
          },
          {
            "at": "50.0%",
            "color": "#836aa1"
          },
          {
            "at": "75.0%",
            "color": "#372352"
          },
          {
            "at": "100%",
            "color": "#190033"
          }
        ]
      },
```

Set permissions for a book, view, or websheet

If you are an administrator, you can set permissions for a book, view, or websheet.

Set permissions for a book, view, or websheet in Planning Analytics Workspace

About this task

When you set permissions, it is important to understand whether you are modifying permissions that are inherited from the folder where the object resides or whether you are creating a new policy of permissions on this object only.

For example, from a book you can set permissions based on the permissions that are inherited from the folder that the book resides in. This approach means that you use **effective permissions**.

Or you can set permissions specifically for the book. This approach means that you create a **policy** of explicit permissions.

Procedure

1. Click the **Reports and Analysis** tile on the **Home** page.
2. Click the **Shared** tab on the **Reports and Analysis** page.
3. Find the book, view, or websheet for which you want to set permissions. Use the filter, search, and sort options on the **Shared** tab to narrow the list of items.
4. Click the **Actions** icon ⋮ for the item on which you want to set permissions.
5. Click **Set permissions**.

   By default, the **Inherit permissions from <parent> content folder** option is selected, indicating that the items inherit permissions from the parent folder. If you leave this option selected, you cannot customize permissions for individual items.
6. Clear the **Inherit permissions from <parent> content folder** option
7. By default, you see all users and groups for which you can set permissions on the item, regardless of the current permissions.

   Click the **Filter** icon ▽ to narrow the view by users and groups or by current permission status.
8. For each user or group for which you want to assign permissions, select one of the following **Permission** options:

   - **View only** - The user or group can view content in the book, view, or websheet and edit values
   - **Edit only** - The user or group can view content in the book, view, or websheet and edit values.
   - **Full control** - The user or group has full administrative control over the book, view, or websheet and can modify and delete the item.

Set permissions for a book in Planning Analytics Workspace Classic

About this task

When you set permissions, it is important to understand whether you are modifying permissions that are inherited from the folder where the object resides or whether you are creating a new policy of permissions on this object only.

For example, from a book you can set permissions based on the permissions that are inherited from the folder that the book resides in. This approach means that you use **effective permissions**.

Or you can set permissions specifically for the book. This approach means that you create a **policy** of explicit permissions.

Procedure

1. On the **Welcome** page, click ⋮ on the tile of the book that you want to set permissions for.
2. Click **Set permissions**.
3. By default, you see **Showing effective permissions**, which means that you are inheriting permissions from the folder where the object is.
   a) Clear **Inherit permissions from folder** and click **Clear all** to reset the permissions and set a policy of permissions for the book.
   b) Click **Inherit permissions from folder** to have the book inherit permissions from the folder that it resides in.

      If you select this option, you can't customize permissions for individual users.
4. Optional: By default, you see all users who you can set permissions for (**All permissions**).
   - To see only users who already have permissions on the book, select **Assigned only** in the filter beside **All permissions**.
   - To see only users who are not already assigned permissions on the book, select **Unassigned only** in the filter beside **All permissions**.
5. By default, you see all users and groups that you can set permissions for (**Users and groups**).
   - To see only users to set permissions for, select **Users only** in the filter beside **Users and groups**.
   - To see only groups to set permissions for, select **Groups only** in the filter beside **Users and groups**.
6. For each user that you want to have access to the book, click the appropriate permission level.
   - **View** - The user can view content in the book but can't edit it.
   - **Edit** - The user can view content in the book and edit values.
   - **Full control** - The user has full administrative control over the book and can modify and delete it.

   You can give a consumer, analyst, modeler, or administrator **View**, **Edit**, or **Full control** access.

Set permissions on the Shared folder and subfolders

If you are an administrator in Planning Analytics Workspace, or a user with full control permissions, you can set user permissions on the Shared folder. Additionally, administrators and other users with required privileges can set user permissions on subfolders within the Shared folder.

Set permissions on the Shared folder and subfolders in Planning Analytics Workspace

Before you begin

Note: It is recommended that the **Everyone** group is assigned the **View** permission on the root of the shared folder.

About this task

Because the Shared folder is the highest level of where you can set permissions, this approach is called setting a **policy** for the Shared folder. This **policy** affects all objects in the Shared folder.

Procedure

1. Click the **Reports and Analysis** tile on the **Home** page.
   a) Click the **Shared folder** tab.
   b) Click the **Set permissions** icon.
   c) On the **Set permissions** dialog box, by default you see all users and groups for which you can set permissions, regardless of the current permissions.

      Click the **Filter** icon ▽ to narrow the view by users and groups or by current permission status.

d) For each user or group for which you want to set permissions, select one of the following **Permission** options:

- **View only** - The user can view content in the folder, but cannot save content to the folder.
- **Edit only** - The user can view and add content to the folder, and can also set permissions on content that the user adds to the folder, but cannot set permissions on the folder itself.
- **Full control** - The user has full control over maintenance of the folder and its contents. For a subfolder anywhere under the Shared folder, the user can move, delete, or rename the subfolder.

e) Click **Save**.

2. To set permissions for any subfolder within the Shared folder:

a) Click the **Reports and Analysis** tile on the **Home** page.

b) Click the **Shared** tab on the **Reports and Analysis** page.

c) Click the **Filter** icon ▽ and select only the **Folder** option.

You can now see all the folders within the Shared folder.

d) Click the **Actions** icon ⋮ for the folder on which you want to set permissions.

e) Click **Set permissions**.

By default, the **Inherit permissions from Shared content folder** option is selected, indicating that the folder inherits permissions from the Shared folder. If you leave this option selected, you cannot customize permissions for individual users or groups.

f) Clear the **Inherit permissions from Shared content folder** option

g) By default, you see all users and groups for which you can set permissions, regardless of the current permissions.

Click the **Filter** icon ▽ to narrow the view by users and groups or by current permission status.

h) For each user or group for which you want to set permissions, select one of the following **Permission** options:

- **View only** - The user can view content in the folder, but cannot save content to the folder.
- **Edit only** - The user can view and add content to the folder, and can also set permissions on content that the user adds to the folder, but cannot set permissions on the folder itself.
- **Full control** - The user has full control over maintenance of the folder and its contents. For a subfolder anywhere under the Shared folder, the user can move, delete, or rename the subfolder.

i) Click **Save**.

Set permissions on the Shared folder and subfolders in Planning Analytics Workspace Classic

Before you begin

Note: It is recommended that the **Everyone** group is assigned the **View** permission on the root of the shared folder.

About this task

Because the Shared folder is the highest level of where you can set permissions, this approach is called setting a **policy** for the Shared folder. This **policy** affects all objects in the Shared folder.

Procedure

1. To set permissions for the Shared folder:

a) Click the **Shared** folder on the Welcome page, then click Folder Permissions.

   Note: The **Folder Permission** icon is available only when the Shared folder is selected.

b) On the **Set permissions** dialog box, by default you see all users who you can set permissions for (**All permissions**).

   - To see only users who already have permissions on the Shared folder, select **Assigned only** in the filter beside **All permissions**.
   - To see only users who are not already assigned permissions, select **Unassigned only** in the filter beside **All permissions**.

c) By default, you see all users and groups that you can set permissions for (**Users and groups**).

   - To see only users to set permissions for, select **Users only** in the filter beside **Users and groups**.
   - To see only groups to set permissions for, select **Groups only** in the filter beside **Users and groups**.

d) Click the user or group that you want to set permissions for, then click one of the following permissions:

   - **View** - The user can view content in the folder, but cannot save content to the folder.
   - **Edit** - The user can view and add content to the folder, and can also set permissions on content that the user adds to the folder, but cannot set permissions on the folder itself.
   - **Full control** - The user has full control over maintenance of the folder and its contents. For a subfolder anywhere under the Shared folder, the user can move, delete, or rename the subfolder.

e) Click **OK**.

2. To set permissions for any subfolder within the Shared folder:

   a) Open the parent folder that contains the subfolder for which you want to set permissions.

   b) On the subfolder tile, click ⋮.

   c) Click **Set permissions** and clear the **Inherit permissions from folder 'parent'** option.

   By default, the **Inherit permissions from folder 'parent'** option is selected, indicating that the subfolder inherits permissions from the parent folder. If you leave this option selected, you cannot customize permissions for individual users.

   d) On the **Set permissions** dialog box, by default you see all users who you can set permissions for (**All permissions**).

   - To see only users who already have permissions on the folder, select **Assigned only** in the filter beside **All permissions**.
   - To see only users who are not already assigned permissions on the folder, select **Unassigned only** in the filter beside **All permissions**.

   e) By default, you see all users and groups that you can set permissions for (**Users and groups**).

   - To see only users to set permissions for, select **Users only** in the filter beside **Users and groups**.
   - To see only groups to set permissions for, select **Groups only** in the filter beside **Users and groups**.

   f) Click the user or group that you want to set permissions for.

   g) Click one of the available permissions: **View**, **Edit**, or **Full control**.

   h) Click **OK**.

Delete chats from a book

If you are an administrator, you can delete the entire chat conversation from a book.

Procedure

1. On the Welcome page, click ⋮ on the tile of the book from which you want to delete the chat conversation.
2. Click **Delete chats**.
3. Click **Delete** again when prompted for confirmation.

Limit access to the Set Editor

If you are an administrator, you can enable or disable access to the Set Editor from a view or from a selector that is created from a dimension tile.

By default, users can access the Set Editor from a dimension tile in a view or from a selector that is created from a dimension tile. When you disable access to the Set Editor, users cannot open the Set Editor, but they can select from members of the current set.

Limit access to the Set Editor Planning Analytics Workspace

Procedure

1. Click a view or a selector in a book.
2. Click the **Properties** icon .
3. Click the **Custom** tab.
4. Click **Set editor**.
5. Enable or disable the **Allow access to Set editor** option.

Limit access to the Set Editor Planning Analytics Workspace Classic

Procedure

1. Click a view or a selector in a book.
2. Click the **Properties** icon , then click **Set Editor**.
3. Clear the **Allow access to Set Editor** option.
4. To restore access to the Set Editor, repeat steps 1 and 2, then select the **Allow access to Set Editor** option.

Unload and reload a cube from memory

You can unload a cube from memory to temporarily reduce RAM consumption or to assist in the development and troubleshooting of rules feeders. Unloading a cube also unloads any views of the cube from memory.

About this task

While a cube is unloaded, any reference to or request for data in the cube will cause the cube to be automatically reloaded, maintaining data availability.

You must be an administrator or modeler to unload or reload cubes.

Procedure

1. Make sure that you are in edit mode.
2. In the Data tree, right-click the cube you want to unload, then click **Unload cube**.
3. Click **OK** when prompted for confirmation.

   To manually reload a cube into memory, right-click the cube, then click **Reload cube**.

Chapter 8. Tutorials

Getting started in IBM Planning Analytics Workspace is easy. To find out what you can achieve, try these tutorials.

Analyzing expenses
This tutorial introduces you to the basic techniques that are needed to analyze expenses in Planning Analytics Workspace:

- Create a book.
- Navigate using the tree.
- Add a view to the book, and resize it.
- Focus on the data that you want to see by creating sets from dimensions.

Creating a map visualization
This tutorial shows you how to create a map visualization from TM1 data.

Creating dimensions and hierarchies
This tutorial shows you how to use multiple dimensions and hierarchies to model data about car sales.

Working with scorecards
This tutorial demonstrates how to work with scorecards and impact diagrams to explore relationships between metrics.

Tutorial: Analyzing expenses in Planning Analytics Workspace

This tutorial demonstrates the steps for analyzing expenses.

Creating a new book

You are going to create a new book to focus on expenses in your business.

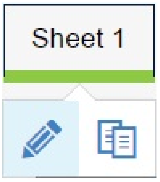
This video illustrates the steps described below:

https://youtu.be/rxNp3lgaUlU

Procedure

1. In a web browser, go to the URL that you have been given and log in.
2. On the Welcome page, click ⊕ and then click **Book from template**.
3. Keep the default selections for the dashboard (Tabbed) and layout (Freeform), and then click **Create**.
4. Rename **Sheet 1** to `Expense input`.

   Tip: Click **Sheet 1** and then click the pencil icon.

5. Click 💾, name the book Expense input, choose whether this book is **Shared** or **Personal**, then click **Save**.

What to do next

Next, add some data to the Expense input sheet.

Creating the expenses plan

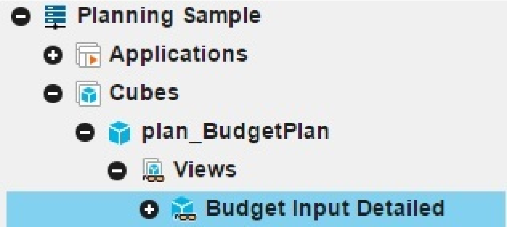
To create the expenses plan, add a view that already exists to the Expense input sheet.

This video illustrates the steps described below:

https://youtu.be/3vBm7LVuKis

Procedure

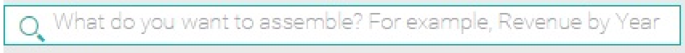
1. Ensure that you are in edit mode.
2. Click + to show the tree.
3. In the tree, expand **Planning Sample** > **Cubes** > **plan_BudgetPlan** > **Views**.



Planning Sample
Applications
Cubes
plan_BudgetPlan
Views
Budget Input Detailed

4. Drag the **Budget Input Detailed** view onto the Expense input sheet.

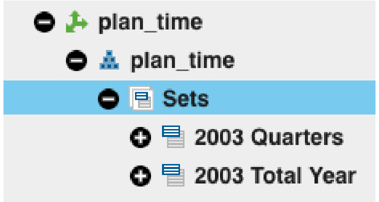
   Tip: You can also find and add this view to the sheet by typing `view budget input detailed` in the intent bar:



What do you want to assemble? For example, Revenue by Year

5. Click the **Budget Input Detailed** view. Resizing handles appear around the edges. ○.
6. Resize the view to make it the full width of the Expense input sheet and half the height, so that you have room to add another view on the sheet.
7. Go to the tree, and expand **Dimensions** > **plan_time** dimension in the tree.

   Tip: You might need to scroll down the tree to find it the dimension. A dimension has this icon: ⤷ and then click ⁂ to see **Sets**, **Levels**, and **Members**
8. Expand **Sets**.



plan_time
plan_time
Sets
2003 Quarters
2003 Total Year

   Sets are selections of members in dimensions that you can use for focusing on smaller groups of members.
9. Find the **Current Desc Year Qtr Month** set and drag it on top of the **plan time** tile on the Expenses input sheet.

   This set replaces the **plan time** dimension with the **Current Desc Year Qtr Month** set.

   Tip: The **plan time** tile turns yellow when the **Current Desc Year Qtr Month** set is in the correct position to replace it.
10. Click the **plan chart of accounts** tile, and then click ✎ to open the set editor.
11. Click the bullet icon ◉ and then click **Children**

◎

.

This option means that when you select a member for insertion into the **Current Set** list, the member's children are also selected.

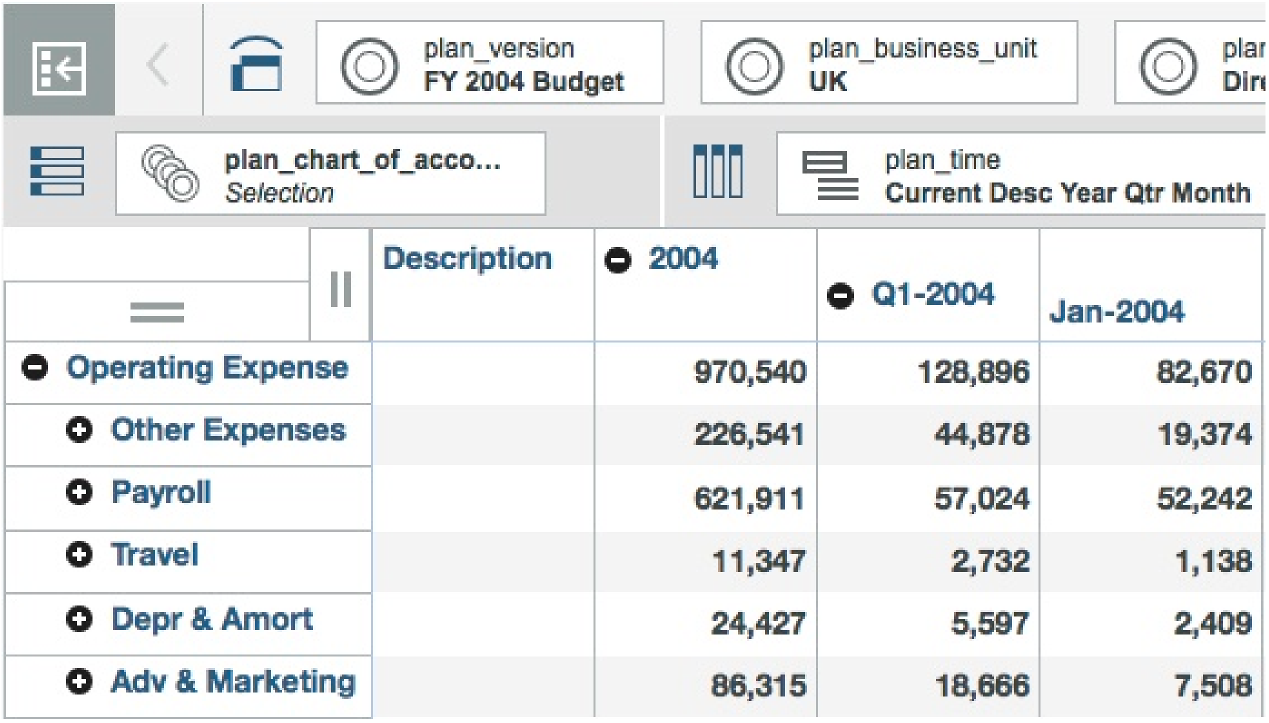
12. Click **Operating Expense** in the **Available Members** pane.
13. Click **Replace and close**.
14. Click ⊕ to expand **Operating Expense**.
15. Click the **plan_department** tile and then click ✎ to open the set editor.
16. Click **Display aliases**

    and select one of the aliases, for example, **Department_English**. Then click **Apply and close**. Notice that the members in **plan_department** change from numbers to text in the language of the alias that you selected.

17. Click the pencil ✎ to leave edit mode.

Results

Your view should look similar to this image (the numbers might be different):



plan_version FY 2004 Budget

plan_business_unit UK

plar Dir

plan_chart_of_acco... *Selection*

plan_time Current Desc Year Qtr Month

	Description	2004	Q1-2004	Jan-2004
Operating Expense		970,540	128,896	82,670
Other Expenses		226,541	44,878	19,374
Payroll		621,911	57,024	52,242
Travel		11,347	2,732	1,138
Depr & Amort		24,427	5,597	2,409
Adv & Marketing		86,315	18,666	7,508

Checking a value

Because many people can use the same sample data, you need to check that a cell value is a specific value before doing the next tasks. This task is for the purposes of the tutorial scenario only, however, it demonstrates some useful techniques.

You can view this video to illustrate the procedure described below:

https://youtu.be/ExTxqgjIJQo

Procedure

1. On the Expense input sheet, double-click the **Travel** member so that you can see more detail.
2. In the view, click ▬ to display the shortcut bar, and then click >_
3. In the snap command bar, type `Germany and Marketing` and press Enter. This changes the plan business unit and plan department selections in the context area to Germany and Marketing.

   To close the snap command bar, click <
4. In the **Travel - Other**, and **Feb-2004** cell, ensure that the value is around `13,937,000`. If it is not this value, type `13,937k` in the cell.
5. In the shortcut bar, click >_ and in the snap command bar, type `UK and Direct` to change the plan business unit and plan department selections back to their previous states. Click ▬ to close the shortcut bar.
6. Right-click the **Travel** row header and select **Drill up**.

   Then, click ⊕ to expand **Operating Expense**.

Planning the advertising and marketing budget

You want to do some planning for the advertising and marketing budget.

This video illustrates the planning process:

https://youtu.be/2HwO59WavVQ

Procedure

1. Double-click the **Adv & Marketing** member to drill down on it.
2. Keep the budget for Advertising and Marketing in Q1 constant by applying a hold. Right-click the cell at the intersection of **Adv & Marketing** and **Q1-2004**, and select **Hold**.
3. You are going to give more weight to the investment in your web site. Click the intersection of the cell for **Web Site**, **Mar-2004**, and change the value to `10000`.

   Tip: You can also type `10k`.

   See that the total value for **Adv & Marketing** stays the same, even though the value for **Web Site** in **Mar-2004** has increased.
4. Double-click the **Web Site**, **Description** cell to add a comment to say what you have done.
5. Release the hold: right-click the intersection of the cell for **Adv & Marketing** and **Q1-2004**, and select **Release hold**.

Creating a visualization to show expenses month by month

Now you'll duplicate the Expense input view so that you have two views on the Expense input sheet.

You'll also modify the new view and change it from an exploration (table) to a stack column visualization. Then you'll add the new visualization to the collection so that it can be easily reused in the same book or in another book.

Procedure

1. You want to investigate **Operating Expense**. Right-click in the Adv & Marketing row header and select **Drill up**.
2. Change what you see in the Expense input view so that you have a global view of expenses.

a) In the view, click to display the shortcut bar, and then click

b) In the snap command bar, type `all business units and all departments`.

3. Click the pencil to go into edit mode.
4. Click a cell to display the shortcut bar, then click the **Duplicate** icon to duplicate the **Expense input** view. Drag the duplicate beneath the original view with the drag handle:

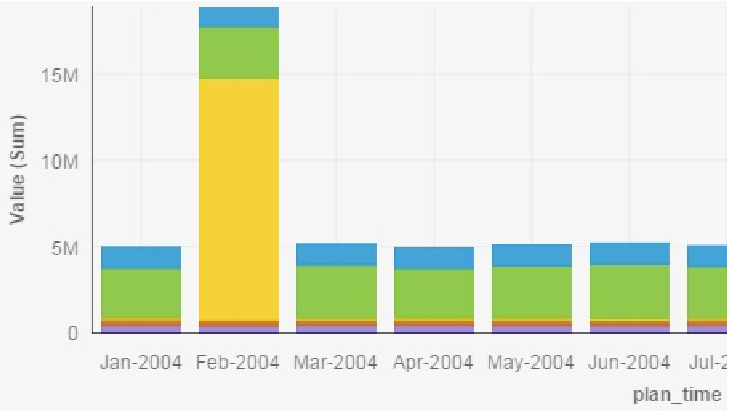
   Tip: You might need to click in the white space and then click on the duplicated chart to select it.
5. You want to see only months, not quarters in the new view. In the tree, go to **Planning Sample** > **Cubes** > **plan_BudgetPlan** > **Dimensions** > **plan_time** > **plan_time** > **Sets**, and drag the **n_level 2004 time** set on top of the **Current Desc Year Qtr Month** tile to replace the dimension.

   Note: If you can't see the **Planning Sample** in the tree, click the up arrow at the top of the tree to go to the previous level in the tree.
6. You don't want to see the total **Operating Expense** in the new view. Right-click **Operating Expense** and select **Hide**.
7. Turn the new view into a visualization. Click the view, click then click from the shortcut bar. Choose the **Stack Column** visualization.
8. You want time to be on the x-axis, so swap the rows and columns around by clicking and clicking

   .

   You now have a stack chart that shows you the operating expenses over the year.
9. Click the **Show/hide overview** icon in the shortcut bar to hide the context area.

   This gives the visualization more space. You can see that travel expenses are very high in February.

10. Select the collections icon in the shortcut bar.

    This adds a link to your visualization to your collection at the top of the browser. Objects saved in Collection can be easily accessed and reused either in the same Planning Analytics Workspace book or in another book.

Tip: Click the Collections icon at the top of the browser to show the Collection.

Analyzing why the cost of travel is high in February

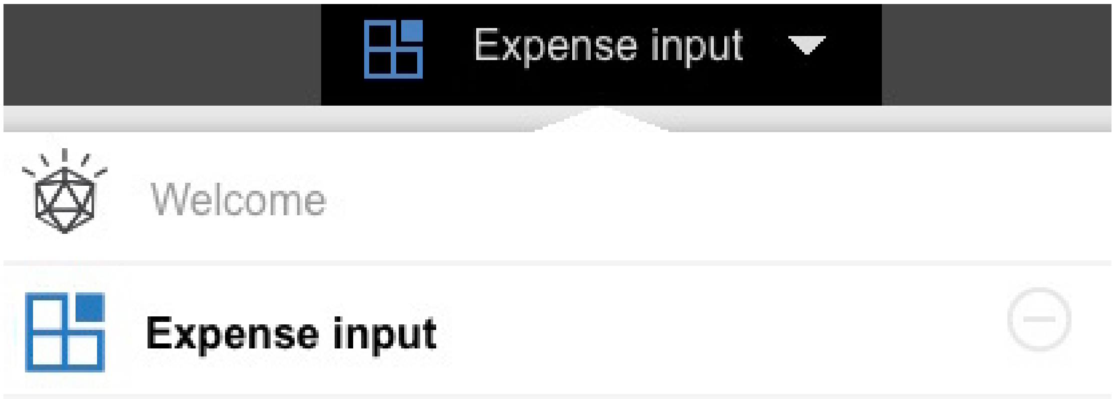
Now it's time to create a new book to analyze why the cost of travel is high in February.

You'll also add a view from the collection, and modify this view to focus on travel in February. You'll use visualizations to figure out where the high cost is, and you'll fix the problem.

Procedure

1. Create a new book that uses the default layout.

   To create a book, go back to the **Welcome** screen by clicking `Expense input` (the name of your book) at the top of your browser, and selecting **Welcome**.

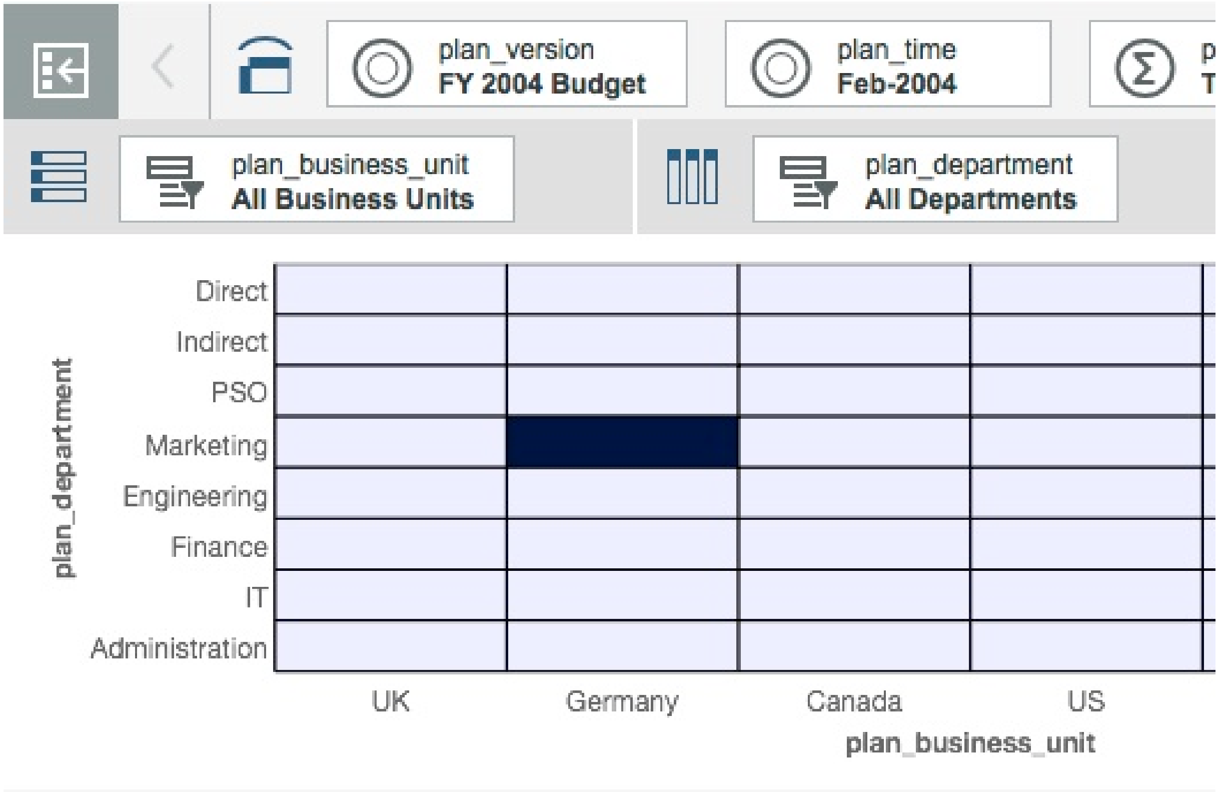
   

   Then click ⊕ and then **Book from template**.
2. Name the new book `Travel Analysis` and click **Create**.
3. Click and select whether the book will be **Shared** or **Personal** and click **Save**.
4. Name the sheet `Travel`.
5. Click − to hide the tree and free up some space.
6. Click the collection icon at the top of the browser, then click **Collection** to show your saved collections, and drag the **Expense input** view onto the Travel sheet.

   The chart highlights expenses. Focusing on February, you want to see the business unit, the organization, and department that the expenses come from.
7. Click the pencil icon ✎ to come out of edit mode.
8. Click the chart and in the toolbar, click the show overview icon ⌄ to show the headings, and then select **Change visualization** from the shortcut bar. Choose the **Exploration** view.
9. You want to change the context around, but stay focused on the month of February. From the rows, drag **Feb-2004** on top of the **plan business unit** tile. You also want to focus on Travel. Drag **Travel** from the column headings onto the **plan department** tile to swap them around.

10. You want to look at the relative travel costs across individual departments and business units, and you don't want to see any totals. To hide the totals, right-click in the column selector and click **Hide totals**. Repeat for the row headings in the row selector

11. Click the chart, and then select **Change visualization** from the toolbar. Select the **Heat** visualization.



plan_version
FY 2004 Budget

plan_time
Feb-2004

plan_business_unit
All Business Units

plan_department
All Departments

12. In the heat visualization, you can see that the travel spend is all in Germany, in the Marketing department. Click that cell and you can see that the value is 13,937,000. Click the UK and Marketing cell and you can see that the value is just 9,971, and in Canada and Marketing, the value is 3,333.

    It looks like the travel spend figure in the Germany Marketing department is a typing error and should be 13,937.

13. You can fix this issue in the **Expense input** book. To return to the book, click the name of your current book, **Travel Analysis** to show the menu, and then select **Expense input**.

14. Double-click **Travel** to drill down on it.

15. Click the chart, click >_ and in the snap command bar, type `Germany and Marketing`. This snap command sets Germany as the plan_business_unit, and Marketing as the plan_department.

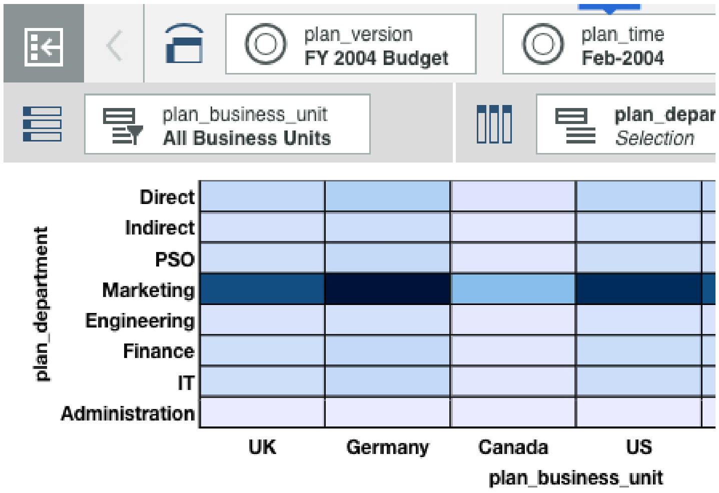
16. Type 13937 in the **Feb-2004** and **Travel** cell.

17. In the stack chart, you should now see that in February there are no anomalies in travel expenses.

18. Switch back to the Travel Analysis book, click , and click

The scale changes, and you can see that the marketing spend on travel across all countries is the highest expenditure.

You can also see that the travel spend in all departments in Canada appears to be lower than everywhere else.



Tutorial: Creating dimensions and hierarchies

This tutorial uses multiple dimensions and hierarchies to model data about car sales.

About this task

Now it's time to create a new view of your sample data, and then create new dimensions and hierarchies that make it easier to model your data. Hopefully, this tutorial demonstrates some of the following benefits of using hierarchies:

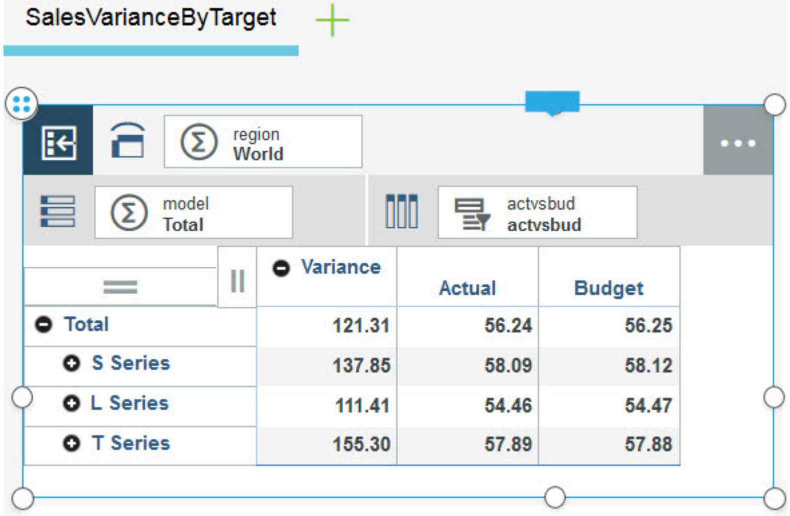
- Hierarchies can improve query performance
- You can turn attributes into hierarchies.
- Modeling attributes as hierarchies instead of dimensions can save memory space. This is because you can have cubes with fewer dimensions if you use hierarchies. Hierarchies act as virtual dimensions.
- Hierarchies give you greater flexibility. A simgle dimension can contain multiple hierarchies, and you can display them in the same view.
- Hierarchies conform to existing standards that already use hierarchies

Note: Think of a dimension as a container of hierarchies instead of a container of members. Now that you can create more than just one hierarchy in a dimension, the hierarchies are the containers of members and the dimension is a container of those hierarchies.

Let's start with the server and look at some data about car sales. In the SData sample in IBM Planning Analytics Local version 2.0.0, the `EnableNewHierarchyCreation` configuration parameter in the tm1s.cfg file is set to `true` so this is a good place to start creating hierarchies.

Procedure

1. Create a book and call your sheet **Sales Variances by Target**.
2. Click ▦.
3. In the content tree, expand **SData** > **SalesCube** > **Dimensions**
4. Drag the **model** dimension onto the rows, the **actvsbud** dimension on to the columns, and the **region** dimension onto the context.

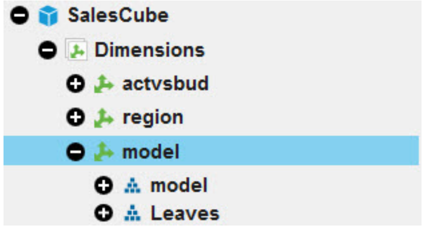


	Variance	Actual	Budget
Total	121.31	56.24	56.25
S Series	137.85	58.09	58.12
L Series	111.41	54.46	54.47
T Series	155.30	57.89	57.88

This view is showing you the variance between actual sales and budgeted sales for each model of car that is broken down by series (S, L, and T Series).

5. In the content tree, expand the **SalesCube** cube, and then expand the **model** dimension.

   You might notice that the **model** dimension has only one hierarchy right now and it is called **model**.



   The **model** hierarchy is the default hierarchy that was created when this dimension was created. You are going to create some more hierarchies and see how they impact the view of your data.

6. Right-click the **model** dimension and click **Create hierarchy**.

   Don't right-click the **model** hierarchy because you can't add a hierarchy under a hierarchy. You can add a hierarchy only to a dimension.

   Tip: Remember, your dimension is a container of hierarchies.

7. Name the new hierarchy **CustomerTarget** and click **Create**.

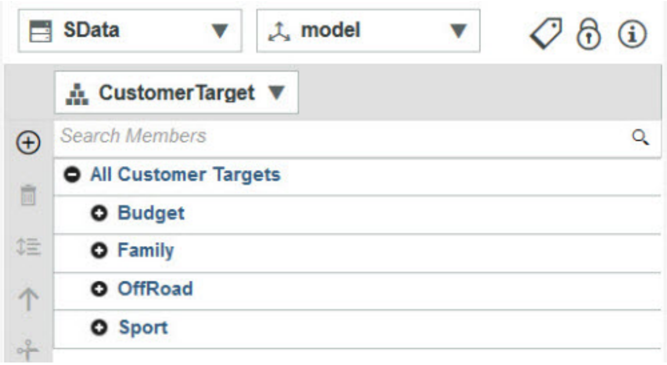
   You are going to use this hierarchy to see a different view of the models of cars that might help you analyze the demographics of your customers better.

   Instead of looking at sales by series, you are going to view car sales based on whether they are targeted at sports car purchasers, off-road drivers, drivers with families, or budget-conscious customers.

8. To make things easier to view, in the **Dimension Editor** of the **CustomerTarget** hierarchy, click ⊕ and create a member that is called **All Customer Targets**.

9. In the hierarchy editor, create four new members in the **CustomerTarget** hierarchy and add them as children of the **All Customer Targets** member.

   - **Budget**
   - **Family**
   - **OffRoad**
   - **Sport**



- All Customer Targets
  - Budget
  - Family
  - OffRoad
  - Sport

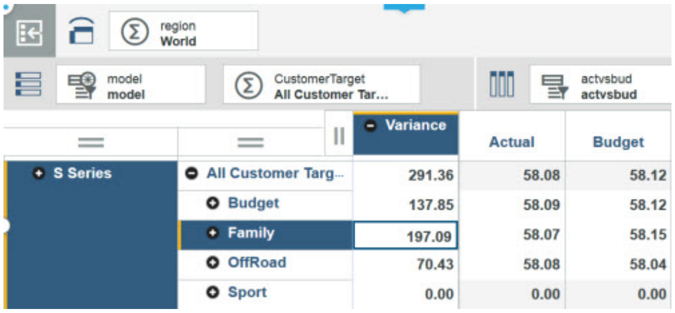
Because your hierarchy has no data about cars, there's nothing useful about this hierarchy yet.

10. Populate your new hierarchy with members. For tips and tricks, see "Data entry" on page 117.
    a) Right-click the **model** hierarchy and click **Edit Hierarchy**.
    b) Search for all the members of the **model** dimension and find all the wagons using the filter

       and copy them by using right-click **Copy member**.
    c) Paste each of the wagons into the **Family** member of the **CustomerTarget** hierarchy by using CTRL+V or CMD+V.

       You can copy and paste multiple members at a time if they are in a continuous selection. You should find 12 members to add.
    d) Find all the convertibles and add them to the hierarchy as **Sport** members of the **CustomerTarget** hierarchy. There should be a total of four convertibles.
    e) Find all the coupes and add them to the hierarchy as **Sport** members of the **CustomerTarget** hierarchy. There should be a total of four coupes.
    f) Add all the 4WD vehicles to the **OffRoad** member of the **CustomerTarget** hierarchy. You should find eight 4WD vehicles to add as members here.
    g) Add all the Sedans to the **Budget** member of the **CustomerTarget** hierarchy. There are 13 sedans to add.

    If you want to, you can right-click the **model** dimension and click **Add as selector widget**. This makes it easier to jump to data points in your view. For more information, see "Create selectors" on page 152.
11. Drag your **CustomerTarget** hierarchy beside your **model** hierarchy in your view.
12. Find the intersection of the **model** and **CustomerTarget** hierarchies to find out the sales variance of S-series family vehicles that were sold in 2015.



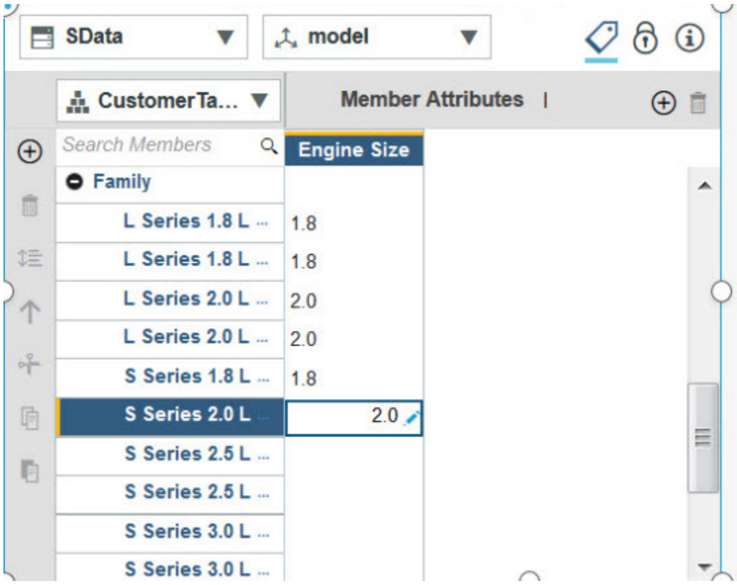
		Variance	Actual	Budget
S Series	All Customer Targ...	291.36	58.08	58.12
	Budget	137.85	58.09	58.12
	Family	197.09	58.07	58.15
	OffRoad	70.43	58.08	58.04
	Sport	0.00	0.00	0.00

13. In the **Dimension Editor** of the **CustomerTarget** hierarchy, click .

    Notice that the members have attributes already because these members exist in the **model** hierarchy also.
14. Right-click all the attributes that contain translated model names and click **Hide**.

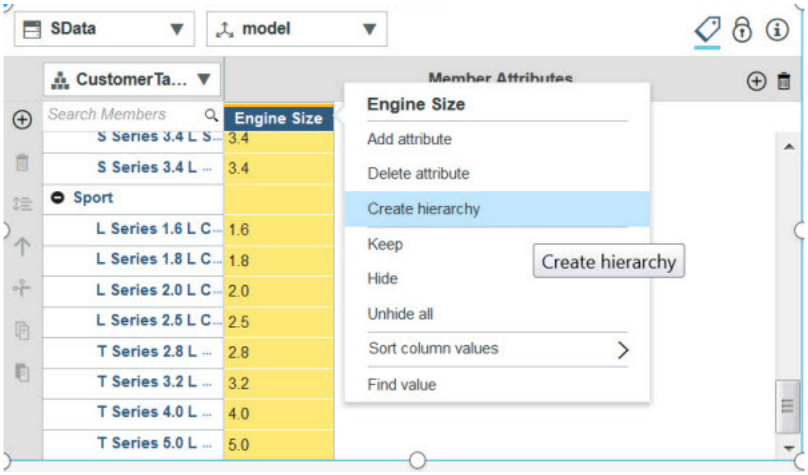
Notice that there is a member attribute called **Engine Size**. Let's populate this attribute with some useful data.

15. For each member, in the **Engine Size** attribute, enter the engine size of the car by using the series name. For example, for the L Series 1.8 L Sedan, enter 1.8 in the **Engine Size** attribute if it is not already filled in.



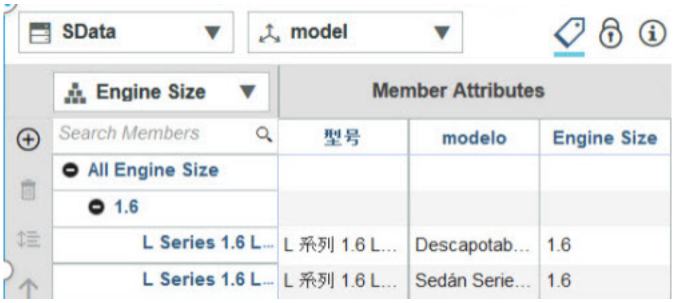
	Engine Size
Family	
L Series 1.8 L ...	1.8
L Series 1.8 L ...	1.8
L Series 2.0 L ...	2.0
L Series 2.0 L ...	2.0
S Series 1.8 L ...	1.8
S Series 2.0 L ...	2.0
S Series 2.5 L ...	
S Series 2.5 L ...	
S Series 3.0 L ...	
S Series 3.0 L ...	

16. Now that you have all the **Engine Size** attributes entered, right-click the **Engine Size** attribute and click **Create hierarchy** and click **OK**.



	Engine Size
S Series 3.4 L S...	3.4
S Series 3.4 L ...	3.4
Sport	
L Series 1.6 L C...	1.6
L Series 1.8 L C...	1.8
L Series 2.0 L C...	2.0
L Series 2.5 L C...	2.5
T Series 2.8 L ...	2.8
T Series 3.2 L ...	3.2
T Series 4.0 L ...	4.0
T Series 5.0 L ...	5.0

A new hierarchy that is named **Engine Size** opens in a **Dimension Editor** widget. This hierarchy contains members that were previously member attributes.



	型号	modelo	Engine Size
All Engine Size			
1.6			
L Series 1.6 L...	L 系列 1.6 L...	Descapotab...	1.6
L Series 1.6 L...	L 系列 1.6 L...	Sedán Serie...	1.6

17. Add the **Engine Size** hierarchy to your view beside the **model** and **CustomerTarget** hierarchies.

What to do next

Find the intersection of the data for all models that are targeted at sports car owners who want at least a 3.2-liter engine. Duplicate your view with the **ProjectedSales** dimension instead of the **actvsbud** dimension and see whether you can figure out which series and engine type is projected to have the most sales to families.

Without hierarchies, this intersection of the data would not be possible.

Tutorial: Working with scorecards

This tutorial demonstrates how to work with scorecards in IBM Planning Analytics Workspace.

You can view this video to follow along in the tutorial:

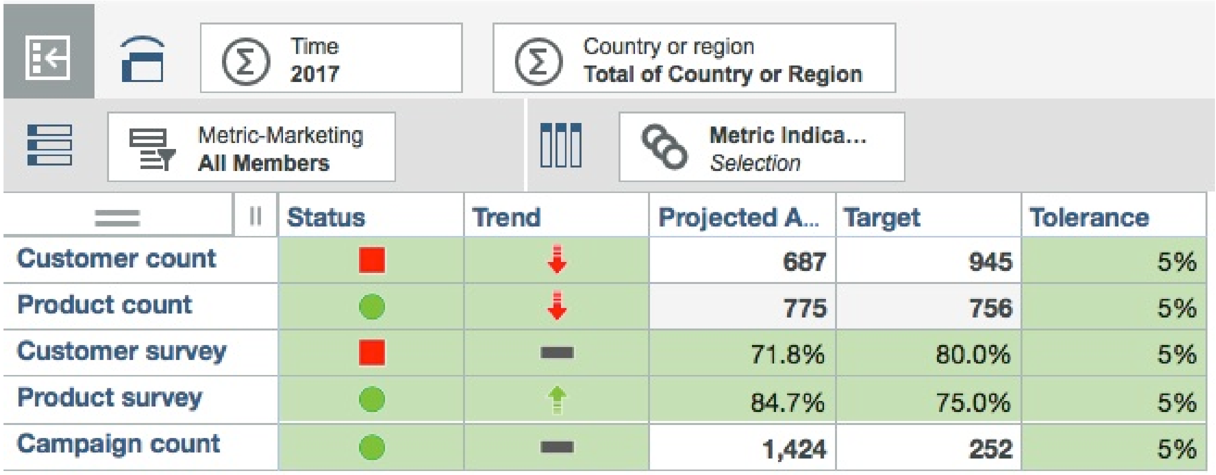
https://youtu.be/AB3tzTeKULQ

About this task

You create a book and add a scorecard and an impact diagram to explore the relationships among metrics. You also set up synchronization between these two visualizations and update the targets for the metrics.

Procedure

1. Create a book with the **Freeform** layout.
2. In the content tree, expand the **GO_Scorecards** database, navigate to **Cubes** > **Metrics cube - Marketing** > **Views**, and drag **View-Marketing** onto the sheet.
3. Click the Metric Indicators tile and click ✏ .
4. Under **Available Members**, hold **CTRL** (Microsoft) or **CMD** (Apple Mac) and select the **Status**, **Trend**, **Projected Actual**, **Target**, and **Tolerance** members.
5. Click **Replace and close** to replace existing members with the selection and close the set editor.



Time
2017

Country or region
Total of Country or Region

Metric-Marketing
All Members

Metric Indica...
Selection

	Status	Trend	Projected A...	Target	Tolerance
Customer count	🟥	↓	687	945	5%
Product count	●	↓	775	756	5%
Customer survey	🟥	−	71.8%	80.0%	5%
Product survey	●	↑	84.7%	75.0%	5%
Campaign count	●	−	1,424	252	5%

6. Click the scorecard view to display the toolbar, and click ⧉ to duplicate the view.

   Tip: To hide the toolbar, click ▼.

7. Using the drag handle ⠿ on the view, drag the new view below the first view.

8. Click the new view to display the toolbar, click change visualization, and select the **Impacts** visualization.
9. Set up synchronization between the scorecard and impact diagram. Click the scorecard view, click , select **Synchronize**, and then **Synchronize dimensions**.
10. Repeat the previous step for the impact diagram. Click **Properties** to close the pane.
11. Modify some targets in the scorecard view and see how they are reflected in the impact view.
    a) In the Time tile of the scorecarding view, select **Feb 2017**.
    b) In the Country or region tile of the synchronization view, select **Canada**.

    Notice how, in the scorecarding view, the Trend icon in the Campaign count row turns from unchanged ▬ to a downward trend ↓. This is also reflected in the impact diagram.

Appendix A. References

Use the following resources to learn more about Planning Analytics Workspace.

Rules Functions

This section contains a complete list of all IBM TM1 rules functions. You can use any of these functions when writing TM1 rules.

Arithmetic Operators in TM1 Rules

The following mathematical operators can be used when constructing TM1 rules.

Operator	Meaning
+ (plus sign)	Addition
- (minus sign)	Subtraction
* (asterisk)	Multiplication
/ (forward slash)	Division by zero using this operator returns an undefined value.
\ (back slash)	Division by zero using this operator returns zero.
^ (caret/circumflex)	Exponentiation

Comparison Operators in TM1 Rules

The comparison operators compare values in the formula portion of a rule calculation statement.

To compare two string values, insert the @ symbol before the comparison operator. For example, `IF ('A' @= 'B',0,1)` yields the number 1.

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
=	Equal to
<>	Not equal to

Logical Operators in TM1 Rules

You can combine expressions in a rules calculation statement using logical operators.

Operator	Meaning	Example
& (ampersand)	AND	(Value1 > 5) & (Value1 < 10) Returns TRUE if the value is greater than 5 and less than 10.
% (percentage sign)	OR	(Value1 > 10) % (Value1 < 5) Returns TRUE if the value is greater than 10 or less than 5.
~ (tilde)	NOT	~(Value1 > 5) Equivalent to (Value1 <= 5)

Attribute Rules Functions

Rules functions that work on attributes.

ATTRN

ATTRN returns a numeric attribute for a specified element of a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATTRN(dimension, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
element	An element of the dimension.

attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.
	Note: : When this function is used in a conditional statement (IF), the statement is the portion containing the condition, not the entire conditional block. After a minor error, execution continues with the next statement. TI processing has no knowledge that it was in a conditional once the minor error is processed, so the next statement is the next line, not the line after the endif.
	To avoid this situation, use variables for any operation that could encounter a minor error and then use the variables in the conditional statement. For example:
	`V1 = CELLGETN('PNLCube', 'fred',`
	`'argentina','Sales','Jan');`
	`IF(V1 = 454);ASCIIOUTPUT`
	`('bug.txt', 'if logic not working`
	` properly');`
	`ENDIF;`

Example

In this example, the function returns the numeric value of the Engine Size attribute of the L Series 1.8L Sedan element in the Model dimension.

```
ATTRN('Model', 'L Series 1.8L Sedan', 'Engine Size')
```

ATTRS

ATTRS returns a string attribute for a specified element of a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATTRS(dimension, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
element	An element of the dimension.
attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Example

In this example, the function returns the string value of the Currency attribute of the 10100 element in the plan_business_unit dimension.

```
ATTRS('plan_business_unit', '10100', 'Currency')
```

CubeATTRN

CubeATTRN returns a numeric attribute for a specified cube.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CubeATTRN(CubeName, AttrName);
```

Argument	Description
CubeName	A valid cube name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the cube.

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Product cube.

```
CubeATTRN('Product', 'Accounting_Code');
```

CubeATTRS

CubeATTRS returns a string attribute for a specified cube.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CubeATTRS(CubeName, AttrName);
```

Argument	Description
CubeName	A valid cube name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the cube.

Example

In this example, the function returns the string value of the Owner attribute of the Product cube.

```
CubeATTRS('Product', 'Owner');
```

DimensionATTRN

DimensionATTRN returns a numeric attribute for a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DimensionATTRN(DimName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Plan_Business_Unit dimension.

```
DimensionATTRN('Plan_Business_Unit', 'Accounting_Code');
```

DimensionATTRS

DimensionATTRS returns a string attribute for a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DimensionATTRS(DimName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the string value of the Manager attribute of the Plan_Business_Unit dimension.

```
DimensionATTRS('Plan_Business_Unit', 'Manager');
```

ElementAttrN

ElementAttrN returns a numeric attribute for a specified element of a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementAttrN(dimension, hierarchy, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	An element of the dimension.

attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.
	Note: : When this function is used in a conditional statement (IF), the statement is the portion containing the condition, not the entire conditional block. After a minor error, execution continues with the next statement. TI processing has no knowledge that it was in a conditional once the minor error is processed, so the next statement is the next line, not the line after the endif.
	To avoid this situation, use variables for any operation that could encounter a minor error and then use the variables in the conditional statement. For example:
	`V1 = CELLGETN('PNLCube', 'fred', 'argentina','Sales','Jan');`
	`IF(V1 = 454);ASCIIOUTPUT ('bug.txt', 'if logic not working properly');`
	`ENDIF;`

Example

In this example, the function returns the numeric value of the Engine Size attribute of the L Series 1.8L Sedan element in the Automobile hierarchy of the Model dimension.

```
ElementAttrN('Model', 'Automobile', 'L Series 1.8L Sedan', 'Engine Size')
```

ElementAttrS

ElementAttrS returns a string attribute for a specified element of a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementAttrS(dimension, hierarchy, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	An element of the dimension.
attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Example

In this example, the function returns the string value of the Currency attribute of the 10100 element in the expense hierarchy of the plan_business_unit dimension.

```
ElementAttrS('plan_business_unit', 'expense', '10100', 'Currency')
```

Consolidation Calculation Rules Functions

The ConsolidatedMax; ConsolidatedMin; ConsolidatedAvg; ConsolidatedCount; and Consolidated CountUnique perform mathematical calculations on consolidations.

ConsolidatedAvg

ConsolidatedAvg calculates the average value in a consolidation and returns that single value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ConsolidatedAvg(flag-value, cube-name, element_1, element_2,… );
```

Arguments

flag-value

The `flag-value` is the **sum** of the following option values:

- 1 - Use weighting when computing the value of consolidated values within the consolidation for which you are determining the average. If this option value is not included in the `flag-value` sum, the raw value of a consolidated element is used.

  The following conditions might affect whether zeros are included in the calculation.

  - If zero is specified as the weighting of some consolidated elements, then the Tm1s.cfg configuration parameter ZeroWeightOptimization=F must be set for these elements to be included in the calculation of the average value in a consolidation. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not included when calculating the average value in a consolidation.
  - If you want cells containing the value zero to be included when calculating the average, UNDEFVALS must be set in the rules for the cube that is specified by the cube-name argument. This ensures that when a zero is assigned to a cell of the cube, an actual zero value is stored in the cell and the zero value is included when calculating the average value in a consolidation.
  - If the rules for the cube that is specified by the cube-name argument include a SKIPCHECK statement, zeros are *always* ignored when calculating the average value in a consolidation. Remove the SKIPCHECK statement from the rule to include zeros in the consolidation average.
- 2 - Ignore zero values. If this value is included in the flag-value sum, zero values will not be included in the calculation of the average value in a consolidation.

There are three valid values for flag-value.

- 1 - Use consolidation weighting when computing the consolidation average.
- 2 - Ignore zero values when computing the consolidation average.
- 3 - Use consolidation weighting **and** ignore zero values when computing the consolidation average.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a TurboIntegrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you can write a rule such as ['Apr']=ConsolidatedAvg(0, '', !actvsbud, '1 Quarter');

element_1, element_2, ...

Dimension element names that define the intersection of the cube containing the consolidation for which you want to determine the average value.

Arguments *element_1* through *element_n* are sequence-sensitive. *element_1* must be an element from the first dimension of the cube, *element_2* must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

In a cube that is called Income Statement with three dimensions that are named Regions, Time, and Income Statement, the Income Statement dimension contains an element that is called Gross Sales for the overall sales number.

To calculate the average sales across all regions in the year 2010, write:

```
ConsolidatedAvg( 1,  'Income Statement', 'All Regions', '2010', 'Gross Sales' );
```

ConsolidateChildren

ConsolidateChildren forces consolidated values to be calculated by summing immediate children along a specified dimension. ConsolidateChildren is useful when intermediate consolidations are calculated by rules and you want a parent consolidation to be calculated by summing the intermediate consolidations rather than by summing the underlying leaf values.

This function is valid in TM1 rules only.

Syntax

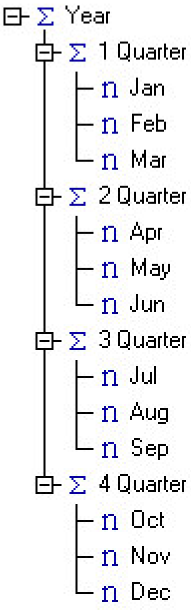
```
ConsolidateChildren(DimName1, DimName2, ...)
```

Argument	Description
DimName1, DimName2, ...	Names of the dimensions along which consolidations will be performed. The function requires at least one DimName argument, and can accept as many DimName arguments as there are dimensions in the cube for which the rule is written.

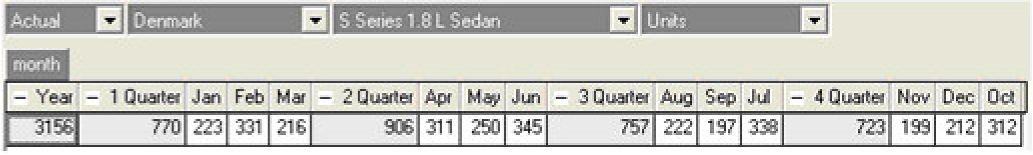
Example

Consider a cube named Sales composed of the dimensions ActVsBud, Region, Model, Account1, and Month.

In this example, the Month dimension is defined as follows:



If no rule is in place for this cube, the value of the Year consolidation is calculated by summing all the underlying leaf values, in this case Jan through Dec. The following image illustrates this consolidation.



Actual | Denmark | S Series 1.8 L Sedan | Units

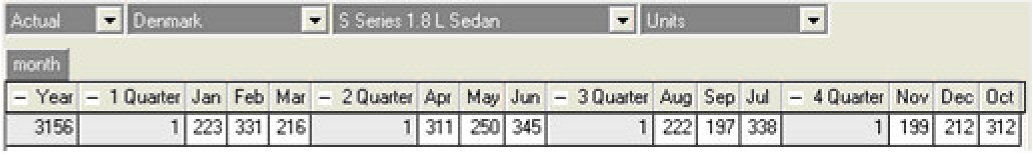
month

− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
3156	770	223	331	216	906	311	250	345	757	222	197	338	723	199	212	312

Now, suppose you create the following rule for this cube, which indicates that all quarterly values should be 1:

```
[{'1 Quarter', '2 Quarter', '3 Quarter', '4 Quarter'}]=1;
```

The result is as follows:



Actual | Denmark | S Series 1.8 L Sedan | Units

month

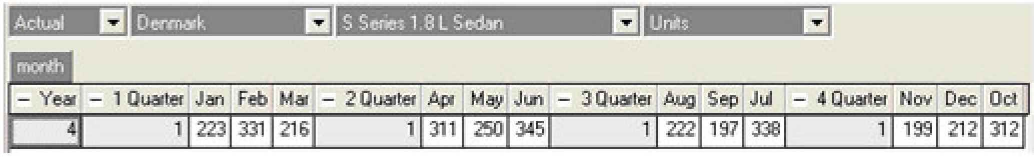
− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
3156	1	223	331	216	1	311	250	345	1	222	197	338	1	199	212	312

In the following image, you can see that quarterly values are indeed calculated by the rule, but the Year consolidation is still calculated by summing all underlying leaf values. If this is not your desired calculation path, you can use the ConsolidateChildren function to force TM1 to calculate the Year consolidation by summing its immediate children, specifically 1 Quarter, 2 Quarter, 3 Quarter, and 4 Quarter.

```
['Year']=ConsolidateChildren('Month');[{'1 Quarter', '2 Quarter', '3 Quarter', '4 Quarter'}]=1;
```

In the rule, the statement `['Year']=ConsolidateChildren('Month')` says that the Year consolidation should be calculated by summing the immediate children of Year in the Month dimension.

The following image shows the result of the `['Year']=ConsolidateChildren('Month')` statement:



− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
4	1	223	331	216	1	311	250	345	1	222	197	338	1	199	212	312

Note that the Year consolidation is now calculated by summing its immediate children.

It's important to remember that for a given consolidation, the ConsolidateChildren function applies only to the *immediate* children of the consolidation.

The ConsolidateChildren function can also be used to specify how consolidations are calculated in multiple dimensions, as in the following example:

Argument	Description
['World','Year']= ConsolidateChildren('Region','Month')	This statement applies the ConsolidateChildren function to both the World and Year consolidations. In this case, World is calculated by summing all its immediate children in the Region dimension, while Year is calculated by summing its immediate children in the Month dimension.

ConsolidatedCount
ConsolidatedCount returns the number of values in a consolidation.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ConsolidatedCount(flag-value, cube-name, element_1, element_2,… );
```

Arguments

flag-value

The `flag-value` is the **sum** of the following option values:

- 1 - Use weighting when computing the value of consolidated values within the consolidation for which you are counting values. If this option value is not included in the `flag-value` sum, the raw value of the consolidated element is used.

  The following conditions might affect whether zeros are included in the calculation.

  – If zero is specified as the weighting of some consolidated elements, then the `Tm1s.cfg` configuration parameter `ZeroWeightOptimization=F` must be set for these elements to be included in the count of values in a consolidation. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not included when counting the number of values in a consolidation.

  - If you want cells containing the value zero to be included when counting the number of values in a consolidation, UNDEFVALS must be set in the rules for the cube that is specified by the cube-name argument. This ensures that when a zero is assigned to a cell of the cube, an actual zero value is stored in the cell and the zero value is included when counting the number of values in a consolidation.
  - If the rules for the cube that is specified by the cube-name argument include a SKIPCHECK statement, zeros are *always* ignored when counting the number of values in a consolidation. Remove the SKIPCHECK statement from the rule to include zeros when counting the number of values in a consolidation.
- 2 - Ignore zero values. If this value is included in the flag-value sum, zero values will not be includedwhen counting the number of values in a consolidation.

There are three valid values for flag-value.

- 1 - Use consolidation weighting when counting the number of values in a consolidation.
- 2 - Ignore zero values when counting the number of values in a consolidation.
- 3 - Use consolidation weighting **and** ignore zero values when counting the number of values in a consolidation.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a TurboIntegrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:['Apr']=ConsolidatedCount(1, '', !actvsbud, '1 Quarter');

element_1, element_2, ...

Dimension element names that define the intersection of the cube containing the consolidation for which you want to count the number of values.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

ConsolidatedCountUnique

ConsolidatedCountUnique counts the number of unique elements for which data points actually exist for the specified consolidation. The unique elements are counted along one dimension of the consolidated cell.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ConsolidatedCountUnique( flag-value, unique-along-dimension-name, cube-name,elem_1,
elem_2, . . . );
```

Arguments

flag-value

The flag-value is the **sum** of the following option values:

- 1 - Use weighting when computing the number of unique elements for which data points actually exist. If this option value is not included in the flag-value sum, the raw values of elements within the consolidation are used.

  The following conditions might affect whether zeros are included in the calculation.

  - If zero is specified as the weighting of some elements, then the Tm1s.cfg configuration parameter ZeroWeightOptimization=F must be set for these elements to be included in the calculation of the number of unique elements. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not included when calculating the number of unique elements.

  - If you want cells containing the value zero to be included when calculating the number of unique elements with actual values, UNDEFVALS must be set in the rules for the cube that is specified by the cube-name argument. This ensures that when a zero is assigned to a cell of the cube, an actual zero value is stored in the cell and the zero value is included when calculating the number of unique elements with actual values.
  - If the rules for the cube that is specified by the cube-name argument include a SKIPCHECK statement, zeros are *always* ignored when calculating the number of unique elements with actual values. Remove the SKIPCHECK statement from the rule to include zeros in the calculation of the number of unique elements with actual values.

- 2 - Ignore zero values. If this value is included in the flag-value sum, zero values will not be included in the calculation of the number of unique elements with actual values.

There are three valid values for flag-value.

- 1 - Use consolidation weighting when computing the number of unique elements with actual values.
- 2 - Ignore zero values when computing the number of unique elements with actual values.
- 3 - Use consolidation weighting **and** ignore zero values when computing the number of unique elements with actual values.

unique-along-dimension-name

The dimension along which unique element entries with real data are to be counted.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a TurboIntegrator process, the cube-name argument can be specified as an empty string to mean the current cube.

element_1, element_2, ...

Dimension element names that define the intersection of the cube which is the consolidated value to be processed.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

In a cube called Income Statement with three dimensions: Regions, Time, and Income Statement, the Income Statement dimension contains an element called Gross Sales for the overall sales number. To count how many regions had some gross sales in the year 2010 write:

```
ConsolidatedCountUnique( 3, 'Regions', 'Income Statement', 'All Regions', '2010', 'Gross
Sales' );
```

This example uses consolidation weighting **and** ignores zero values when computing the number of unique elements with actual values.

ConsolidatedMax

ConsolidatedMax calculates the maximum value in a consolidation and returns that single value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ConsolidatedMax(flag-value, cube-name, element_1, element_2,… );
```

Arguments

flag-value

The `flag-value` is the **sum** of the following option values:

- 1 - Use weighting when computing the value of consolidated values within the consolidation for which you are determining the maximum. If this option value is not included in the `flag-value` sum, the raw value of the consolidated element is used.

  The following conditions might affect whether zeros are included in the calculation.

  - If zero is specified as the weighting of some consolidated elements, then the `Tm1s.cfg` configuration parameter `ZeroWeightOptimization=F` must be set for these elements to be included in the calculation of the maximum value in a consolidation. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not included when calculating the maximum value in a consolidation.
  - If you want cells containing the value zero to be included when calculating the average, `UNDEFVALS` must be set in the rules for the cube that is specified by the `cube-name` argument. This ensures that when a zero is assigned to a cell of the cube, an actual zero value is stored in the cell and the zero value is included when calculating the maximum value in a consolidation.
  - If the rules for the cube that is specified by the `cube-name` argument include a `SKIPCHECK` statement, zeros are *always* ignored when calculating the maximum value in a consolidation. Remove the SKIPCHECK statement from the rule to include zeros in the calculation of the maximum value.
- 2 - Ignore zero values. If this value is included in the `flag-value` sum, zero values will not be included in the calculation of the maximum value in a consolidation.

There are three valid values for `flag-value`.

- 1 - Use consolidation weighting when computing the maximum value in a consolidation.
- 2 - Ignore zero values when computing the maximum value in a consolidation.
- 3 - Use consolidation weighting **and** ignore zero values when computing the maximum value in a consolidation.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a TurboIntegrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedMax( 1, '', !actvsbud, '1 Quarter' );`

element_1, element_2, ...

Dimension element names that define the intersection of the cube containing the consolidation for which you want to determine the maximum value.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

Consider a cube called Income Statement with three dimensions, "Area", "Time", and "Income Statement". The Income Statement dimension contains an element "Gross Sales" for the overall sales number.

To calculate the maximum sales across all regions in the year 2010 use:

```
ConsolidatedMax( 1,  'Income Statement', 'All Regions', '2010', 'Gross Sales' );
```

ConsolidatedMin

ConsolidatedMin calculates the minimum value in a consolidation and returns that single value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ConsolidatedMin(flag-value, cube-name, element_1, element_2,… );
```

Arguments

flag-value

The `flag-value` is the **sum** of the following option values:

- 1 - Use weighting when computing the value of consolidated values within the consolidation for which you are determining the minimum. If this option value is not included in the `flag-value` sum, the raw value of the consolidated element is used.

  The following conditions might affect whether zeros are included in the calculation.

  - If zero is specified as the weighting of some consolidated elements, then the `Tm1s.cfg` configuration parameter `ZeroWeightOptimization=F` must be set for these elements to be included in the calculation of the minimum value in a consolidation. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not included when calculating the minimum value in a consolidation.
  - If you want cells containing the value zero to be included when calculating the average, `UNDEFVALS` must be set in the rules for the cube that is specified by the `cube-name` argument. This ensures that when a zero is assigned to a cell of the cube, an actual zero value is stored in the cell and the zero value is included when calculating the minimum value in a consolidation.
  - If the rules for the cube that is specified by the `cube-name` argument include a `SKIPCHECK` statement, zeros are *always* ignored when calculating the minimum value in a consolidation. Remove the SKIPCHECK statement from the rule to include zeros in the calculation of the minimum value.
- 2 - Ignore zero values. If this value is included in the `flag-value` sum, zero values will not be included in the calculation of the minimum value in a consolidation.

There are three valid values for `flag-value`.

- 1 - Use consolidation weighting when computing the minimum value in a consolidation.
- 2 - Ignore zero values when computing the minimum value in a consolidation.
- 3 - Use consolidation weighting **and** ignore zero values when computing the minimum value in a consolidation.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a TurboIntegrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedMin( 1, '', !actvsbud, '1 Quarter' );`

element_1, element_2, ...

Dimension element names that define the intersection of the cube containing the consolidation for which you want to determine the minimum value.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Cube Data Rules Functions

Rules functions that work on cube data.

CellValueN

CellValueN returns the numeric value of the specified element(s) in a cube. This is a TM1 rules function, valid in TM1 rules only. Use of this function in a TurboIntegrator process will result in an error.

For dimensions not among the element parameters, coordinates are retrieved from the rule target (the cell being retrieved and triggering rule evaluation). The function behavior is analogous to the intra-cube reference expression (e.g. ['Measures':'Count']), as used on the rule's right-side.

The element parameters may be specified in any order, and for CellValueN, multiple elements from the same dimension (but different hierarchies of the dimension) may be specified. Since the elements list is not required to be in cube dimension order, it is necessary to dimension-qualify all element parameters. Element parameters from multi-hierarchy dimensions must also be hierarchy-qualified.

Syntax

```
CellValueN(cube, element1,..., elementN);
```

Argument	Description
cube	Name of the cube.
elementN	Element name that defines the cell. A minimum of one element must be specified.

Example

```
CellValueS('ForecastCube', 'Products':'ProductsByChannel':'Channel2', 'Measures':'Count');
```

This example returns the numeric value of the specified cell. The Products dimension has multiple hierarchies while the Measures dimension has one hierarchy.

The intra-cube reference is restricted to literal parameters, while CellValueN is not. This behavior is analogous to the DB() rules function. The element parameters may be specified using string-valued expressions. For example, the above Products element parameter could be specified as:

```
'Products' : 'ProductsByChannel' : AttrS( … )
```

Unlike DB() and the intra-cube reference expression, CellValueN element parameters must be either dimension-qualified, or dimension and hierarchy qualified.

CellValueS

CellValueS returns the string value of the specified element(s) in a cube. This is a TM1 rules function, valid in TM1 rules only. Use of this function in a TurboIntegrator process will result in an error.

For dimensions not among the element parameters, coordinates are retrieved from the rule target (the cell being retrieved and triggering rule evaluation). The function behavior is analogous to the intra-cube reference expression (e.g. ['Measures':'Count']), as used on the rule's right-side.

The element parameters may be specified in any order, and for CellValueS, multiple elements from the same dimension (but different hierarchies of the dimension) may be specified. Since the elements list is not required to be in cube dimension order, it is necessary to dimension-qualify all element parameters. Element parameters from multi-hierarchy dimensions must also be hierarchy-qualified.

Syntax

```
CellValueS(cube, element1,..., elementN);
```

Argument	Description
cube	Name of the cube.
elementN	Element name that defines the cell. A minimum of one element must be specified.

Example

```
CellValueS('ForecastCube', 'Products':'ProductsByChannel':'Channel2', 'Measures':'Location');
```

This example returns the string value of the specified cell. The Products dimension has multiple hierarchies while the Measures dimension has one hierarchy.

The intra-cube reference is restricted to literal parameters, while CellValueS is not. This behavior is analogous to the DB() rules function. The element parameters may be specified using string-valued expressions. For example, the above Products element parameter could be specified as:

```
'Products' : 'ProductsByChannel' : AttrS( … )
```

Unlike DB() and the intra-cube reference expression, CellValueS element parameters must be either dimension-qualified, or dimension and hierarchy qualified.

DB

DB returns a value from a cube in a TM1 database. DB returns a numeric value if used in a numeric expression and a string value if used in a string expression.

DB is a TM1 rules function, valid in TM1 rules only. Use of this function in a TurboIntegrator process causes an error.

Syntax

```
DB(cube, e1, e2, [...e256])
```

Parameters

cube

The name of the cube from which to retrieve the value.

e1,...en

Dimension element names that define the intersection containing the value to be retrieved.

Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on.

Example

In this example, Budget is the cube name, and the function returns the value at the intersection of California, 15" Flat Panel Monitors, Net Sales, and January.

```
DB('Budget', 'California', '15" Flat Panel Monitors', 'Net Sales', 'January')
```

When used to reference multi-hierarchy dimensions, you must specify the particular hierarchy. In this example, the Category2 element exists in the ByCategory hierarchy of the ProductsCube dimension.

```
DB('ProductsCube', 'ByCategory':'Category2',...)
```

ISLEAF

ISLEAF returns 1 if a specified cell is a leaf cell (identified solely by leaf/simple elements). If the specified cell is identified by any consolidated elements, the function returns 0.

This function is valid in TM1 rules only.

The ISLEAF function cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
ISLEAF
```

Arguments

None.

Example

You can use ISLEAF in an IF statement to test if a current cell is a leaf cell. For example:

```
[]=IF((ISLEAF=1),TrueStatement, FalseStatement);
```

Executes the TrueStatement if the current cell is a leaf cell, otherwise it executes the FalseStatement.

ISUNDEFINEDCELLVALUE

ISUNDEFINEDCELLVALUE compares the passed value to the default numeric cube value, which is influenced by the presence of the UNDEFVALS declaration in that cube's rule. The function returns 1 if the passed value equals the cube's default value, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

`ISUNDEFINEDCELLVALUE(`*`TestValue, <Cube>`*`)`

Arguments

Argument	**Description**
TestValue	The numerical value to compare against the cube's default value.
Cube	An optional String argument that specifies the cube whose default value should be compared. When ISUNDEFINEDCELLVALUE is used in a rule, the cube is assumed to be the subject cube unless otherwise specified. When used in a TI, a cube should be specified. If the cube is omitted in a TI, or is not valid when specified, 0 will be used for comparison.

Example

`ISUNDEFINEDCELLVALUE(TestValue)` returns 1 when TestValue is the special undefined value and is used in the rule of a cube with UNDEFVALS declared.

UNDEF

UNDEF returns the undefined value. This function can be used to prevent datafrom being stored in a cube based on a logical test.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
UNDEF
```

Arguments

None.

Example

UNDEF returns the undefined value.

UNDEFINEDCELLVALUE

UNDEFINEDCELLVALUE returns the default numeric cube value, which is influenced by the presence of the UNDEFVALS declaration in that cube's rule.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

`UNDEFINEDCELLVALUE(<Cube>)`

Arguments

Argument	Description
Cube	An optional String argument that specifies the cube whose default value should be returned. When UNDEFINEDCELLVALUE is used in a rule, the cube is assumed to be the subject cube unless otherwise specified. When used in a TI process, a cube should be specified. If the cube is omitted in a TI process, or is not valid when specified, 0 will be returned.

Example

UNDEFINEDCELLVALUE returns 0 when used in the rule of a cube without UNDEFVALS declared, or when used in a TI process.

UNDEFINEDCELLVALUE returns the special undefined value when used in the rule of a cube with UNDEFVALS declared.

`UNDEFINEDCELLVALUE('ExampleCube')` returns the default value of 'ExampleCube', or 0 if 'ExampleCube' does not exist.

UNDEFVALS

Putting UNDEFVALS in the rules for a cube changes the default value for the cube from zero to a *special undefined* value. Like other rules functions, UNDEFVALS applies only to the cube associated with the rule in which the function appears.

This is a TM1 rules function, valid only in TM1 rules.

Use of UNDEFVALS has ramifications regarding how data is stored in the cube and retrieved.

- **Data Storage**

  For a cube without UNDEFVALS in the rules, the default value is zero. If an attempt is made to store a zero in a cell of the cube, that storage request is ignored, as this is a redundant attempt to store the default value, and it would needlessly consume memory space. Similarly, if a cell already contains a value and the value is deleted, nothing is stored in the cell.

  If however the cube has UNDEFVALS defined in the rules, this makes the default value a *special undefined* value. Now when a zero is stored in a cell of a cube, it is actually stored, just like any other non-zero value.

  The *special undefined* value is only a run-time value, returned from requests for cell values. It is never stored in an actual cell in memory, and is never written to disk. Including UNDEFVALS in the rule for a cube has no effect on memory usage or disk storage, except for cells that actually contain zero as a value. When UNDEFVALS is included in the rule for a cube, zero values in that cube will consume memory space and will be written to disk, just like any other data value. If UNDEFVALS is not specified, zero value cells are not stored in memory nor are they written to disk.

- **Data Retrieval**

  For a cube without UNDEFVALS in the rules, the default value is zero. When a cell is retrieved, and there is no value currently stored for that value in the cube, a value of zero (as the default value) is returned. This means that an application cannot tell whether a cell actually exists and contains zero as the cell value, or whether the cell does not exist (as can be the case with sparse data).

  If however the cube has UNDEFVALS defined in the rules, this make the default value a *special undefined* value. In this case, when a non-existent cell is retrieved, the value retrieved will be this *special undefined* value. This can be used to distinguish a cell that does not exist (*special undefined* returned) from a cell that exists, but whose value is zero (zero returned). Any client written to run against TM1, which can encounter a cube with UNDEFVALS set, must be prepared to handle a cell value of this *special undefined* rather than a zero. A client can detect whether a value returned from TM1 is this *special undefined* value with the `TM1ValIsUndefined` API function. For details on the `TM1ValIsUndefined` API function, see the *TM1 API* documentation.

  Note: This *special undefined* value is not the value returned by the UNDEF() TurboIntegrator function. The value returned by UNDEF() is an undefined value used for such things as an attempt to divide by zero, or take the logarithm of an illegal number, etc.

In TurboIntegrator, for normal arithmetic operations (+, -, *, /, \, ^) and normal arithmetic comparisons (<, >, >=, <=, =, <>), the *special undefined* value is treated as a zero. Because of this, the following code does not work:

```
NoCellVal = UndefinedCellValue( 'cube-name' );
If ( vv = NoCellVal );
```

In this comparison, `NoCellVal`, which is the *special undefined* value for an UNDEFVALS cube, is treated as a zero. This means the comparison is really `If ( vv = 0 )`.

In TurboIntegrator you must use the IsUndefinedCellValue to test if a cell value is the *special undefined* value. For example:

```
 vv = CellGetN( 'cube-name', elements-list);
if ( IsUndefinedCellValue( vv, 'cube-name' ) = 1 );
#the cells does not exist
cell_does_not_exist = 1;
else;
#cell exists
cell_does_not_exist = 0;
Endif;
```

Syntax

```
UNDEFVALS
```

Arguments

None.

Date and Time Rules Functions

Rules functions that work with dates and time.

DATE

DATE returns the date string in *yy-mm-dd* or *yyyy-mm-dd* format for a given serial number.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DATE(SerialNumber, ReturnFourDigitYear)
```

Argument	Description
SerialNumber	A date expressed in serial format.
ReturnFourDigitYear	An optional Boolean argument that determines whether the DATE function returns a string using two- or four-digit notation for the year. If ReturnFourDigitYear is true, the function returns date falling within the range of Jan. 1, 1960 and Dec. 31, 9999, using four-digit notation for the year. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ReturnFourDigitYear is false, or if this optional argument is omitted from the DATE function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059, using two-digit notation for the year. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ReturnFourDigitYear is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, DATE uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

DATE(13947) returns `98-03-09`.

DATE(13947, 1) returns `1998-03-09`.

DATES

DATES returns a date string, in the form 'yy-mm-dd' or 'yyyy-mm-dd', corresponding to a given year, month, and day.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DATES(year, month, day)
```

Argument	Description
year	A year, expressed in either yy or yyyy format.
month	A month, expressed in mm format.
day	A day, expressed in dd format.

Example

DATES(98, 2, 10) returns '98-02-10'.

DATES(1998, 2, 10) returns '1998-02-10'.

DAY

DAY returns a numeric value for the day in a given date string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DAY(DateString)
```

Argument	Description
DateString	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

DAY('02-05-25') returns 25.

DAYNO

DAYNO returns the serial date number corresponding to a given date string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Note: DAYNO can return serial dates for date strings starting at January 1, 1960 (dates string 1960-01-01 or 60-01-01). For dates after December 31, 2059, you use a four digit year in the date string. For example, the date string for January 5, 2061 would be 2061-01-05.

Syntax

```
DAYNO('DateString')
```

Argument	Description
DateString	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

DAYNO('98-03-09') returns 13947.

MONTH

MONTH returns a numeric value for the month in a given date string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MONTH(date)
```

Argument	Description
date	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

MONTH('02-05-25') returns 5.

NOW

NOW returns the current date/time value in serial number format.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
NOW
```

Arguments

None.

Example

NOW returns the current date/time value in serial number format.

TIME

TIME returns a string, in HH:MM format, representing the system time on the TM1 server.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIME
```

Arguments

None.

Example

Given a system time of 9:33 AM, TIME returns the string '09:33'.

Given a system time of 9:33 PM, TIME returns the string '21:33'.

TIMST

TIMST returns a formatted date/time string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIMST(datetime, format, ExtendedYears)
```

Argument	Modifier/ Description
datetime	A date/time serial number. The integer part of the number specifies the date, and the decimal part specifies the time within the day. Day number 0 corresponds to '60-01-01'. Negative numbers correspond to prior years. Years in the 21st Century, up to 2059, are represented by years 00 through 59. An hour is 1/24th of a day, a minute 1/60th of an hour, and a second 1/60th of a minute.
format	A string that formats the result of the function. All the characters in the format argument appear in the result, except for the following characters, which return date/time component values:
	\y the last two digits of the year (97, 98, etc.)
	\Y the four digits of the year (1997, 1998, etc.)
	\m the two digits of the month (01 through 12)
	\M the abbreviation of the month (JAN, FEB, etc.)
	\d the two digits of the day (01 through 31)
	\D the digit of the day (1 through 31)
	\h the hour in military time (00 through 23)
	\H the standard hour (1 through 12)
	\i the minute (00 through 59)
	\s the second (00 through 59)

Argument	Modifier/ Description
	\p a.m. or p.m.
ExtendedYears	This optional Boolean parameter specifies whether the function returns a date falling within the range 1960 - 2059 or 1960 - 9999. If ExtendedYears is true, the function returns a date falling within the range of Jan. 1, 1960 and Dec. 31, 9999. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ExtendedYears is false, or if this optional argument is omitted from the TIMST function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ExtendedYears is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TIMST uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

TIMST(366.0000, '\M \D, \Y') returns 'JAN 1, 1961'.

TIMST(366.5000, '\H\p \imin\ssec') returns '12p.m. 00min00sec'.

TIMST(366.1000, 'On \M \D, \Y at \H\p \imin\ssec') returns 'On JAN 1, 1961 at 2a.m. 24min00sec'.

TIMST(11111.1100, 'On \M \D, \Y at \H\p \imin\ssec') returns 'On JUN 3,1990 at 2a.m. 38min24sec'.

TIMVL

TIMVL returns the numeric value of a component (year, month, etc.) of a date/time value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIMVL(datetime, type, ExtendedYears)
```

Argument	Modifier and Description
datetime	A date and time serial number. The integer part of the number specifies the date, and the decimal part specifies the time within the day. Day number 0 corresponds to '60-01-01.' Negative numbers correspond to prior years. Years in the 21st Century, up to 2059, are represented by years 00 through 59. An hour is 1/24th of a day, a minute 1/60th of an hour, and a second 1/60th of a minute.
type	A character that specifies the type of component to be extracted. The following are valid type arguments:
	Y year value (1997, 1998, etc.)
	M month value (1 through 12)
	D day value (1 through 31)
	H hour value (0 through 23)
	I minute value (00 through 59)
	S second value (00 through 59)

Argument	Modifier and Description
ExtendedYears	This optional Boolean parameter specifies whether the function returns a date falling within the range 1960 - 2059 or 1960 - 9999. If ExtendedYears is true, the function returns a date falling within the range of Jan. 1, 1960 and Dec. 31, 9999. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ExtendedYears is false, or if this optional argument is omitted from the TIMVL function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ExtendedYears is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TIMVL uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

TIMVL(11111.1100, 'Y') returns 1990.

TIMVL(11111.1100, 'H') returns 2.

TODAY

TODAY returns the current date in yy-mm-dd format.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TODAY(<ReturnFourDigitYear>)
```

Argument	Description
ReturnFourDigitYear	An optional Boolean argument that determines whether the TODAY function returns a string using two- or four-digit notation for the year. If ReturnFourDigitYear is true, the function returns date falling within the range of Jan. 1, 1960 and Dec. 31, 9999, using four-digit notation for the year. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ReturnFourDigitYear is false, or if this optional argument is omitted from the TODAY function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059, using two-digit notation for the year. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ReturnFourDigitYear is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TODAY uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

P1=TODAY(1) returns a data string in YYYY-MM-DD format such as 2009-06-05.

P1=TODAY(0) returns a date string in YY-MM-DD format such as 09-06-05

YEAR

YEAR returns a numeric value for the year in a given date string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
YEAR(date)
```

Argument	Description
date	A date string in YY-MM-DD format.

Example

YEAR('02-05-25') returns 2.

Dimension Information Rules Functions

Rules functions that manage dimension information.

DIMIX

DIMIX returns the index number of an element within a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMIX(server_name:dimension, element)
```

Argument	Description
dimension	A valid dimension name qualified by the server name.
element	The name of an element within the dimension. If the element is not a member of the dimension specified, the function returns 0.

Example

Brazil has an index value of three in the Region dimension. The example returns 3.

```
DIMIX('planning_sample:Region','Brazil')
```

DIMNM

DIMNM returns the element of a dimension that corresponds to the index argument.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMNM(server_name:dimension, index)
```

Argument	Description
dimension	A valid dimension name qualified by the server name.
index	A value less than or equal to the number of elements in the dimension. If this argument is less than 1, or greater than the number of elements in the dimension, the function returns 0.

Example

This example returns 'Belgium', which is the element within the Region dimension with an index value of 2.

```
DIMNM(planning_sample:'Region',2)
```

DIMSIZ

DIMSIZ returns the number of elements within a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMSIZ(dimension)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.

Example

If the dimension Accounts contains 19 elements, the example returns the value 19.

```
DIMSIZ('Accounts')
```

DNEXT

DNEXT returns the element name that follows the element specified as an argument to the function.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DNEXT(dimension, element)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

If the Location dimension contains the ordered elements California, Oregon, and Washington, the example returns Washington.

```
DNEXT("Location","Oregon")
```

DNLEV

DNLEV returns the number levels in a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

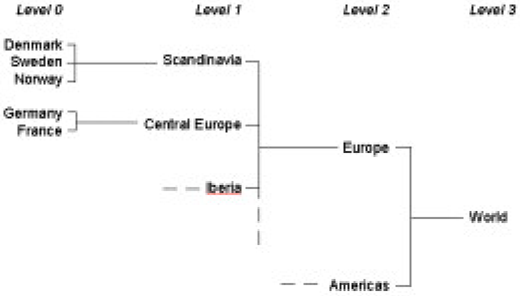
```
DNLEV(dimension)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.

Example

```
DNLEV('Region')
```

In the Region dimension, the various nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3).



There are four levels in the Region dimension, so the example returns the value 4.

DTYPE

DTYPE returns information about the element type of a specified element. DTYPE returns N if the element is a numeric element, S if the element is a string element, and C if the element is a consolidated element.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DTYPE(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

The element Europe is a consolidated element of the Region dimension, so the example returns C.

```
DTYPE('Region','Europe')
```

TABDIM

TABDIM returns the dimension name that corresponds to the index argument.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TABDIM(cube, index)
```

Argument	Description
cube	A valid cube name.
index	A positive value less than or equal to the total number of dimensions in the cube.

Example

The cube SalesCube contains five dimensions: account1, actvsbud, model, month, and region. The example returns model, the third dimension of SalesCube.

```
TABDIM('SalesCube',3)
```

Element Information Rules Functions

Rules functions that manage element information.

ELCOMP

ELCOMP returns the name of a child of a consolidated element in a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELCOMP(dimension, element, position)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension.
position	A positive value less than or equal to the total number of children in the specified element.

Example

In the dimension Region, the consolidated element Central Europe is a consolidation of the children France and Germany. Germany is in the second position in this consolidation. Accordingly, the example returns Germany.

```
ELCOMP('Region','Central Europe',2)
```

ELCOMPN

ELCOMPN returns the number of components in a specified element. If the element argument is not a consolidated element, the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELCOMPN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension.

Example

In the Region dimension, the element Scandinavia is a consolidation of three elements. The example returns 3.

```
ELCOMPN('Region','Scandinavia')
```

ElementComponent

ElementComponent returns the name of a child of a consolidated element in a specified dimension. If the element argument is not a consolidated element, the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementComponent(dimension, hierarchy, element, position)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of a consolidated element within the dimension.
position	A positive value less than or equal to the total number of children in the specified element.

Example

In the dimension Region, the consolidated element Central Europe is a consolidation of the children France and Germany. Germany is in the second position in this consolidation. Accordingly, the example returns Germany.

```
ElementComponent('Region', 'Europe', 'Central Europe', 2)
```

ElementComponentCount

ElementComponentCount returns the number of components in a specified element. If the element argument is not a consolidated element, the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementComponentCount(dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of a consolidated element within the dimension.

Example

In the Region dimension, the element Scandinavia is a consolidation of three elements. The example returns 3.

```
ElementComponentCount('Region', '', 'Scandinavia')
```

ElementCount

ElementCount returns the number of elements within a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementCount(dimension, hierarchy)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.
hierarchy	The name of the hierarchy within the dimension.

Example

If the Receivables hierarchy in the Accounts dimension contains 19 elements, the example returns the value 19.

```
ElementCount('Accounts', 'Receivables')
```

ElementFirst

ElementFirst returns the first element of a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementFirst(server_name:dimension, hierarchy)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.

Example

If the North America hierarchy of the Location dimension contains the ordered elements California, Oregon, and Washington, the example returns California.

```
ElementFirst("planning_sample:Location", "North America")
```

ElementIndex

ElementIndex returns the index number of an element within a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementIndex(server_name:dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name qualified by the server name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension. If the element is not a member of the dimension specified, the function returns 0.

Example

Brazil has an index value of three in the Region dimension. The example returns 3.

```
ElementIndex('planning_sample:Region', 'South America', 'Brazil')
```

ElementIsAncestor

ElementIsAncestor determines whether element1 is an ancestor of element2 in the specified dimension. The function returns 1 if element1 is an ancestor of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementIsAncestor(dimension, hierarchy, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

In the Western hierarchy of the Region dimension, the element Europe is an ancestor of Germany. The example returns 1.

```
ElementIsAncestor('Region', 'Western', 'Europe', 'Germany')
```

ElementIsComponent

ElementIsComponent determines whether element1 is a child of element2 in the specified dimension. The function returns 1 if element1 is a child of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementIsComponent(dimension, hierarchy, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

In the dimension Region, the element Central Europe is a consolidation of two elements, Germany and France. The example returns 1.

Note: this function returns 1 only for immediate children. In the above example, Germany is a child of Central Europe. Further, Central Europe is a child of Europe.

```
ElementIsComponent('Region', 'Countries', 'Germany', 'Central Europe')
```

However, because the function returns 1 only for immediate children, the following example returns 0:

```
ElementIsComponent('Region', 'Countries'', 'Germany', 'Europe')
```

ElementIsParent

ElementIsParent determines whether element1 is a parent of element2 in the specified dimension. The function returns 1 if element1 is a parent of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementIsParent(dimension, hierarchy, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element1	The name of an element within the dimension.

Argument	Description
element2	The name of an element within the dimension.

Example

In the dimension Region, the consolidated element Central Europe is the parent of both Germany and France. Accordingly, the example returns 1.

Note: this function returns 1 only for immediate parents. In the above example, Europe is a parent of Central Europe. Further, Central Europe is a parent of Germany.

```
ElementIsParent('Region', 'Countries', 'Central Europe', 'Germany')
```

However, because Europe is not an immediate parent of Germany, the following example returns 0:

```
ElementIsParent('Region', 'Countries', 'Europe', 'Germany')
```

ElementLevel

ElementLevel returns the level of an element within a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

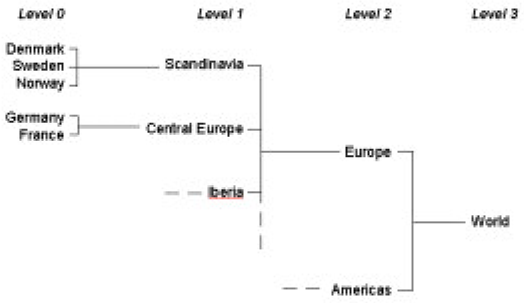
```
ElementLevel(dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension.

Example

```
ElementLevel('Region','Countries', 'Europe')
```

In the Region dimension, individual nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3). The example returns 2, as Europe is a Level 2 element.



ElementName

ElementName returns the element of a dimension that corresponds to the index argument.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementName(dimension, hierarchy, index)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
index	A value less than or equal to the number of elements in the dimension. If this argument is less than 1, or greater than the number of elements in the dimension, the function returns 0.

Example

This example returns 'Belgium', which is the element within the Countries hierarchy of the Region dimension with an index value of 2.

```
ElementName('Region', 'Countries', 2)
```

ElementNext

ElementNext returns the element name that follows the element specified as an argument to the function.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementNext(dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

If the Location dimension contains the ordered elements California, Oregon, and Washington, the example returns Washington.

```
ElementNext("Location","Cities", "Oregon")
```

ElementParent

ElementParent returns the parent of an element in a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementParent(dimension, hierarchy, element, index)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension.
index	A positive value less than or equal to the total number of consolidated elements (parents) that use the element argument as a child.

Example

In the dimension Model, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. In the structure of the Model dimension, Total Wagons is defined first, Total 4WD is defined second.

```
ElementParent('Model', 'Automobile', 'Wagon 4WD', 2)
```

The example returns Total 4WD, as this is the second instance of a parent to Wagon 4WD within the Model dimension.

ElementParentCount

ElementParentCount returns the number of parents of an element in a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementParentCount(dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension.

Example

In the Model dimension, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. The function returns 2.

```
ElementParentCount('Model', 'Automobile', 'Wagon 4WD')
```

ElementType

ElementType returns information about the element type of a specified element. ElementType returns N if the element is a numeric element, S if the element is a string element, and C if the element is a consolidated element.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementType(dimension, hierarchy, element)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element	The name of an element within the dimension.

Example

The element Europe is a consolidated element of the Region dimension, so the example returns C.

```
ElementType('Region', 'Countries', 'Europe')
```

ElementWeight

ElementWeight returns the weight of a child in a consolidated element.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ElementWeight(dimension, hierarchy, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
hierarchy	The name of the hierarchy within the dimension.
element1	The name of a consolidated element within the dimension.
element2	The name of a child of the consolidated element.

Example

The element Variable Costs, which is a child of Gross margin, has a weight of -1. The following example returns -1.

```
ElementWeight('Account1', 'SubAccount1', 'Gross margin', 'Variable Costs')
```

ELISANC

ELISANC determines whether element1 is an ancestor of element2 in the specified dimension. The function returns 1 if element1 is an ancestor of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELISANC(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

In the dimension Region, the element Europe is an ancestor of Germany. The example returns 1.

```
ELISANC('Region', 'Europe', 'Germany')
```

ELISCOMP

ELISCOMP determines whether element1 is a child of element2 in the specified dimension. The function returns 1 if element1 is a child of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELISCOMP(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

In the dimension Region, the element Central Europe is a consolidation of two elements, Germany and France. The following example returns 1.

Note: this function returns 1 only for immediate children. In this example, Germany is a child of Central Europe. Further, Central Europe is a child of Europe.

```
ELISCOMP('Region','Germany','Central Europe')
```

However, because the function returns 1 only for immediate children, the following example returns 0:

```
ELISCOMP('Region','Germany','Europe')
```

ELISPAR

ELISPAR determines whether element1 is a parent of element2 in the specified dimension. The function returns 1 if element1 is a parent of element2, otherwise the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELISPAR(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

In the dimension Region, the consolidated element Central Europe is the parent of both Germany and France. Accordingly, the following example returns 1.

Note: this function returns 1 only for immediate parents. In this example, Europe is a parent of Central Europe. Further, Central Europe is a parent of Germany.

```
ELISPAR('Region','Central Europe','Germany')
```

However, because Europe is not an immediate parent of Germany, the following example returns 0:

```
ELISPAR('Region','Europe','Germany')
```

ELLEV

ELLEV returns the level of an element within a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

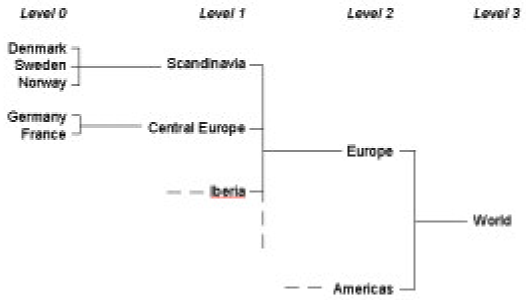
```
ELLEV(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

```
ELLEV('Region','Europe')
```

In the Region dimension, individual nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3). The example returns 2, as Europe is a Level 2 element.



ELPAR

ELPAR returns the parent of an element in a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELPAR(dimension, element, index)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.
index	A positive value less than or equal to the total number of consolidated elements (parents) that use the element argument as a child.

Example

In the dimension Model, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. In the structure of the Model dimension, Total Wagons is defined first, Total 4WD is defined second.

```
ELPAR('Model','Wagon 4WD',2)
```

The example returns Total 4WD, as this is the second instance of a parent to Wagon 4WD within the Model dimension.

ELPARN

ELPARN returns the number of parents of an element in a specified dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELPARN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

In the Model dimension, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. The function returns 2.

```
ELPARN('Model','Wagon 4WD')
```

ELWEIGHT

ELWEIGHT returns the weight of a child in a consolidated element.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELWEIGHT(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of a consolidated element within the dimension.
element2	The name of a child of the consolidated element.

Example

The element Variable Costs, which is a child of Gross margin, has a weight of -1.

The following example returns -1.

```
ELWEIGHT('Account1','Gross margin','Variable Costs')
```

LevelCount

LevelCount returns the number levels in a dimension.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

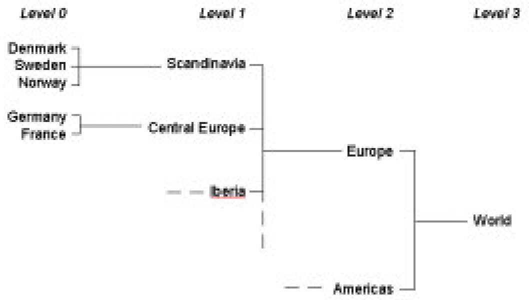
```
LevelCount(dimension, hierarchy)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.
hierarchy	The name of the hierarchy within the dimension.

Example

```
LevelCount('Region', 'Countries')
```

In the Region dimension, the various nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3).



There are four levels in the Region dimension, so the example returns the value 4.

Financial Rules Functions

Rules functions used to manage financial information.

FV

FV returns the value of an annuity at the time of the last payment. An annuity is a series of payments made at equal intervals of time. Payments are assumed to be made at the end of each period.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
FV(payment, interest, periods)
```

Argument	Description
payment	The amount of the payment made per period.
interest	The interest rate paid per period.
periods	The number of periods in the annuity.

Example

This example returns the value of an annuity at the end of 5 years, with payments of $1,000 per year at 14% interest.

```
FV(1000, .14, 5)
```

PAYMT

PAYMT returns the payment amount of an annuity based on a given initial value or principal, an interest rate, and a number of periods. An annuity is a series of payments made at equal intervals of time.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
PAYMT(principal, interest, periods)
```

Argument	Description
principal	The present value, or the total amount that a series of future payments is worth now.
interest	The interest rate paid per period.

Argument	Description
periods	The number of periods in the annuity. Payments are assumed to be made at the end of each period.

Example

This example returns the payment on a 5-year annuity that is paid yearly, with a principal of $100,000 at 14% interest.

```
PAYMT(100000, .14, 5)
```

PV

PV returns the initial or principal value of an annuity.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
PV(payment, interest, periods)
```

Argument	Description
payment	The amount of the payment made.
interest	The interest rate paid per period.
periods	The number of periods in the annuity. Payments are assumed to be made at the end of each period.

Example

This example returns the principal value of an annuity with 5 yearly payments of $1,000 at 14% interest.

```
PV(1000, .14, 5)
```

Hierarchy Rules Functions

Functions to manage hierarchies in rules.

Hierarchy

If there is only one hierarchy included in the supplied dimension, Hierarchy returns the name of the hierarchy. Otherwise, it returns an empty string. Hierarchy is valid in TM1 rules only.

With the introduction of support for multiple hierarchies in TM1, it is necessary to identify which hierarchies are in context when multiple hierarchies are being used.

Hierarchy cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
Hierarchy (DimName);
```

Argument	Description
DimName	A valid dimension name.

Example

This example returns 'Quarter', which is the only hierarchy in the Quarter dimension.

```
Hierarchy ('Quarter');
```

HierarchyCount

HierarchyCount returns the number of hierarchies in the supplied dimension. HierarchyCount is valid in TM1 rules only.

HierarchyCount cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
HierarchyCount (DimName);
```

Argument	Description
DimName	A valid dimension name.

Example

This example returns 3, which is the number of hierarchies in the model dimension.

```
HierarchyCount ('model');
```

HierarchyIndex

HierarchyIndex returns a 1-based index if the hierarchy is in the supplied dimension, 0 otherwise. HierarchyIndex is valid in TM1 rules only.

HierarchyIndex cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
HierarchyIndex (DimName, HierName);
```

Argument	Description
DimName	A valid dimension name.
HierName	A valid hierarchy name that you want to find the index position of in *DimName*.

Example

This example returns 3, which is the index position of the CustomerTarget hierarchy in the model dimension.

```
HierarchyIndex ('model', 'CustomerTarget');
```

HierarchyN

HierarchyN returns the name of the hierarchy at a specified position in the supplied dimension and an empty string if the index is out of scope. HierarchyN is valid in TM1 rules only.

HierarchyN cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
HierarchyN (DimName, index);
```

Argument	Description
DimName	A valid dimension name.
index	A value less than or equal to the number of hierarchies in the dimension. If this argument is less than 1, or greater than the number of hierarchies in the dimension, the function returns 0.

Example

This example returns 'CustomerTarget', which is the third hierarchy in the model dimension.

```
HierarchyN ('model', 3);
```

Logical Rules Functions

Logical operators to use in rules.

CONTINUE

When included as part of a rules expression, CONTINUE allows a subsequent rule with the same area definition to be executed. Normally, TM1 only executes the first rule encountered for a given area.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CONTINUE
```

Arguments

None.

Example

```
['Jan']= if(!region @= 'Argentina',10,CONTINUE);
```

```
['Jan']=20;
```

In this example, all cells identified by January and Argentina are assigned a value of 10. Cells identified by Jan and any other Region element are assigned a value of 20.

IF

IF returns one value if a logical expression you specify is TRUE and another value if it is FALSE.

This function is valid in TM1 rules only.

TurboIntegrator uses its own IF function that is capable of evaluating multiple logical expressions.

Syntax

```
IF(expression, true_value, false_value)
```

Argument	Description
expression	Any value or expression that can be evaluated to TRUE or FALSE.
true_value	The value that is returned if expression is TRUE.
false_value	The value that is returned if expression is FALSE.

Example

IF(1<2, 4, 5) returns 4.

IF(1>2, 'ABC', 'DEF') returns 'DEF'.

STET

The STET function cancels the effect of a rule for a particular element.

This is a TM1 rules function, valid only in TM1 rules. This function cannot be used in TurboIntegrator processes.

Syntax

```
STET
```

Arguments

None.

Example

In this example, the rule dictates that the value for Sales is always 100, except for the intersection of Sales and the element France from the Region dimension.

```
['Sales'] = IF(!Region @= 'France',STET, 100);
```

Mathematical Rules Functions

Mathematical operators to use in rules.

ABS

ABS returns the absolute value of a number.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ABS(x)
```

Argument	Description
x	The number for which you want to find the absolute value.

Example

ABS(-1.2) returns 1.2

ACOS

ACOS returns the angle, in radians, whose cosine is x.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ACOS(x)
```

Argument	Description
x	The cosine of the angle you want to find. x must be between -1 and 1; otherwise the function returns an error.

Example

ACOS(0) returns 1.5708.

ASIN

ASIN returns the angle, in radians, whose sine is x.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ASIN(x)
```

Argument	Description
x	The sine of the angle you want to find. x must be between -1 and 1; otherwise the function returns an error.

Example

ASIN(1) returns 1.5708.

ATAN

ATAN returns the angle, in radians, whose tangent is x. The result is between -pi/2 and +pi/2.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATAN(x)
```

Argument	Description
x	The tangent of the angle you want to find.

Example

ATAN(1) returns .7854.

COS

COS returns the cosine of an angle expressed in radians.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
COS(x)
```

Argument	Description
x	An angle, expressed in radians, for which you want to find the cosine.

Example

COS(0) returns 1.

EXP

EXP returns the natural anti-log of a number.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
EXP(x)
```

Argument	Description
x	A number for which you want to find the natural anti-log.

Example

EXP(1) returns 2.71828.

INT

INT returns the largest integer that is less than or equal to a specified value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
INT(x)
```

Argument	Description
x	A numeric value.

Example

INT(5.6) returns 5.

INT(-5.6) returns -6.

ISUND

ISUND returns 1 if a specified value is undefined; otherwise it returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ISUND(x)
```

Argument	Description
x	A number or expression.

Example

ISUND(5.2) returns 0.

ISUND(1/0) returns 1.

LN

LN returns the natural logarithm (base e) of a number.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LN(x)
```

Argument	Description
x	A positive number. The function returns an error if x is negative or zero.

Example

LN(10) returns 2.302585093.

LOG

LOG returns the base 10 logarithm of a positive number.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LOG(x)
```

Argument	Description
x	A positive number. The function returns an error if x is negative or zero.

Example

LOG(10) returns 1.

MAX

MAX returns the largest number in a pair of values.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MAX(num1, num2)
```

Argument	Description
num1	The first in a pair of values.
num2	The second in a pair of values.

Example

MAX(10, 3) returns 10.

MIN

MIN returns the smallest number in a pair of values.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MIN(num1, num2)
```

Argument	Description
num1	The first in a pair of values.
num2	The second in a pair of values.

Example

MIN(10, 3) returns 3.

MOD

MOD returns the remainder of dividing a number by a divisor.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MOD(number, divisor)
```

Argument	Description
number	The number for which you want to find the remainder.
divisor	The value by which the number argument is divided.

Example

MOD(10, 3) returns 1.

RAND

RAND generates a random number that is uniformly distributed between 0 and 1. The random number generator is seeded when TM1 is loaded.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
RAND.
```

Arguments

None.

Example

RAND generates a random number that is uniformly distributed between 0 and 1

ROUND

ROUND rounds a given number to the nearest integer. Rounding can be done in a variety of valid ways.

This function is valid in both TM1 rules and TurboIntegrator processes.

The most basic form of rounding is to replace an arbitrary number by an integer. There are many ways of rounding a number y to an integer q.

The most common ones are:

- **Round to nearest**

  q is the integer that is closest to y (see "**Round away from zero**" for tie-breaking rules).
- **Round towards zero** (or truncate)

  q is the integer part of y, without its fraction digits.
- **Round down** (or take the floor)

  q is the largest integer that does not exceed y.
- **Round up** (or take the ceiling)

  q is the smallest integer that is not less than y.
- **Round away from zero**

  If y is an integer, q is y; else q is the integer that is closest to 0 and is such that y is between 0 and q.

TurboIntegrator essentially uses the **Round down** method of *floor(x + .5)*. Microsoft Excel uses the **Round to nearest** method. This can result in different integers depending on whether you are using a TurboIntegrator process or working in Excel.

Syntax

```
ROUND(number)
```

Argument	Description
number	The number you want to round.

Example

ROUND(1.46) returns 1.

ROUNDP

ROUNDP rounds a given number at a specified decimal precision.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ROUNDP(number, decimal)
```

Argument	Description
number	The number you want to round.
decimal	The decimal precision at which to apply the rounding. If this argument is positive, the function rounds the specified number of digits to the right of the decimal point. If this argument is negative, the function rounds the specified number of digits to the left of the decimal point. The decimal argument must be between -15 and 15, inclusive.

Example

ROUNDP(1.46, 1) returns 1.5.

ROUNDP(1.466, 2) returns 1.47.

ROUNDP(234.56, -1) returns 230.00.

ROUNDP(234.56, 0) returns 235.00.

SIGN

SIGN determines if a number is positive, negative, or zero. The function returns 1 if the number is positive, -1 if the number is negative, and 0 if the number is zero.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SIGN(number)
```

Argument	Description
number	A number.

Example

SIGN(-2.5) returns -1.

SIN

SIN returns the sine of a given angle.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SIN(x)
```

Argument	Description
x	A value, expressed in radians, for which you want the sine.

Example

SIN(1.5708) returns 1.

SQRT

SQRT returns the square root of a given value.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SQRT(x)
```

Argument	Description
x	Any positive value. SQRT returns an error if x is negative.

Example

SQRT(16) returns 4.

TAN

TAN returns the tangent of a given angle.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TAN(x)
```

Argument	Description
x	A value, expressed in radians, for which you want the tangent.

Example

TAN(0) returns 0.

TAN(.7854) returns 1.

Text Rules Functions

Rules to manage text in rules.

CAPIT

CAPIT applies initial capitalization to every word in a string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CAPIT(string)
```

Argument	Description
string	A text string.

Example

CAPIT('first quarter sales') returns 'First Quarter Sales'.

CHAR

CHAR returns the character identified by a given ASCII numeric code.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CHAR(number)
```

Argument	Description
number	An ASCII code number. This number must be between 1 and 255, inclusive.

Example

CHAR(100) returns 'd'.

CODE

CODE returns the ASCII numeric code for a specified character within a string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CODE(string, location)
```

Argument	Description
string	A text string.
location	A number specifying the character within the string for which you want to find the ASCII code value.

Example

CODE('321', 2) returns 50.

CODE('End', 3) returns 100.

DELET

DELET returns the result of deleting a specified number of characters from a specified starting point within a string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DELET(string, start, number)
```

Argument	Description
string	A text string.
start	The character at which to begin deletion.
number	The number of characters to delete.

Example

DELET('payment', 3, 3) returns 'pant'.

FILL

FILL repeats a given string as necessary to return a string of a specified length.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
FILL(string, length)
```

Argument	Description
string	A text string. This string is repeated as necessary to achieve the specified length.
length	The length of the string you want the function to return.

Example

`FILL('-', 5)` returns `'-----'`.

`FILL('ab', 5)` returns `'ababa'`.

INSRT

INSRT inserts one string into another string at a specified insertion point.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
INSRT(string1, string2, location)
```

Argument	Description
string1	A text string.
string2	A text string.
location	The character in string2 at which you want to insert string1. The function inserts string1 into string2 immediately prior to the character you specify.

Example

INSRT('ABC', 'DEF', 2) returns 'DABCEF'.

LONG

LONG returns the length of a string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LONG(string)
```

Argument	Description
string	A text string.

Example

LONG('Sales') returns 5.

LOWER

LOWER converts all upper case characters in a string to lower case.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LOWER(string)
```

Argument	Description
string	A text string.

Example

LOWER('First Quarter Sales') returns 'first quarter sales'.

NUMBR

NUMBR converts a string to a number. The string passed to the NUMBR function must use. (period) as the decimal separator and , (comma) as the thousand separator. Any other decimal/thousand separators will cause incorrect results.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
NUMBR(string)
```

Argument	Description
string	The string you want to convert to a number. All characters other than '0' through '9', '+', '-', '.', and 'E' are ignored.

Example

NUMBR('-5.6') returns -5.6.

NUMBR('-5A. B6C') returns -5.6.

SCAN

SCAN returns a number indicating the starting location of the first occurrence of a specified substring within a string. If the substring does not occur in the given string, the function returns 0.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SCAN(substring, string)
```

Argument	Description
substring	The substring you are trying to locate.
string	The string within which you are searching for the substring.

The arguments to this function are case-sensitive. The capitalization used in the **`substring`** argument must exactly match the capitalization used in the **`string`** argument for the function to return a non-zero value.

Example

`SCAN('scribe', 'described')` returns 3.

However, `SCAN('Scribe', 'described')` returns 0, because the case in the **`substring`** argument (`Scribe`) does not match the case in the **`string`** argument (`described`).

STR

STR converts a floating point number to a string representing the value in decimal notation.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
STR(number, length, decimal)
```

Argument	Description
number	The floating point number being converted to a string. This number can contain a positive or negative sign. This number can contain decimal places.

Argument	Description
length	The desired count of characters in the string representation, including sign, separators, decimal, or decimal places. The **length** argument value should be a positive number greater than "0". If the **length** argument value is "0" or a negative number, the function returns an empty string. If the count of digits in the number is less than the **length** argument value, the function inserts leading blank spaces to attain this length after inserting sign, separators, decimal, or decimal places. If the count of digits in the whole number exceeds the **length** argument value **and** the **decimal** argument value is "0", the function truncates the whole number to attain this length. If the count of digits in the number exceeds the **length** argument value **and** the **decimal** argument value is greater than "0", the function preserves the whole number and uses the specified decimal places.
decimal	The number of decimal places to include in the function result. If this parameter is "0", a decimal point is not included. If the number specified has more decimal places than the **decimal** argument, the function result is rounded.

All arguments are required and you cannot pass empty argument values.

Note: There is a limitation when using STR with large floating point values. If the count of whole number digits in the **number** argument value exceeds the **length** argument value by more than 5, the function returns an empty string. For example, STR(14723017.2245, 4, 2) returns "14723017.22". The whole number portion of the **number** argument value has 8 digits, which is **not** more than 5 greater than the **length** argument value of 4 (8<5+4). However, STR(14723017.2245, 2, 2) returns an empty string, because the whole number portion of the **number** argument value has 8 digits, which **is** more than 5 greater than the **length** argument value of 2 (8>5+2)

Examples

Function call	Number	Length	Decimal	Result
STR(3.14159, 6, 2)	3.14159	6	2	" 3.14"
STR(-3.14159, 6, 0)	-3.14159	6	0	" -3"
STR(3.14159, 5, 3)	3.14159	5	3	"3.142"
STR(1000000, 4, 0)	1000000	4	0	"1000"

Function call	Number	Length	Decimal	Result
				Note that the number is truncated.
STR(1000000, 4, 2)	1000000	4	2	"1000000.00" Note that the number is not truncated because decimal is specified.
STR(10, 2, 4)	10	2	4	"10.0000"
STR(120536.74391, 8, 0)	120536.74391	8	0	" 120536" The result includes left padding of two spaces to attain the specified length of 8.
STR(120536.74391, 5, 0)	120536.74391	5	5	"12053" The count of digits in the whole number exceeds the **`length`** argument value **and** the **`decimal`** argument value is "0", so the function truncates the whole number to attain the specified length of 5.

SUBST

SUBST returns a substring of a given string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SUBST(string, beginning, length)
```

Argument	Description
string	The string from which you want to extract the substring.
beginning	The character at which the substring begins.
length	The length of the substring.

Example

SUBST('Retirement', 3, 4) returns 'tire'.

TRIM

TRIM returns the result of trimming any leading and trailing blanks from a string.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TRIM(string)
```

Argument	Description
string	A text string.

Example

TRIM(' First Quarter ') returns 'First Quarter'.

UPPER

UPPER converts a text string to upper case.

This function is valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
UPPER(string)
```

Argument	Description
string	A text string.

Example

UPPER('First Quarter Results') returns FIRST QUARTER RESULTS.

Miscellaneous Rules Functions

Rules functions not found in other categories.

FEEDERS

When you use a SKIPCHECK declaration to restore the sparse consolidation in a TM1 rule, you must also ensure that all rules-derived cells are identified by feeder statements. To do this, insert a FEEDERS declaration immediately following all rules statements:

```
FEEDERS;
```

Immediately following the FEEDERS declaration you should create feeders statements that identify the rules-derived cells in the cube.

For a complete discussion of TM1 rules, including sparse consolidation and the creation of feeders, please refer to *TM1 Rules*.

FEEDSTRINGS

Rule-generated string values are not displayed when a view is zero-suppressed unless the string resides in a cell that is fed. To enable feeding of string cells, insert the FEEDSTRINGS declaration as the first line of your rule.

```
FEEDSTRINGS;
```

Once this declaration is in place, you can set up feeders for string cells in a cube view, and rely on the string to be available to other rules even if the view is zero-suppressed. Statements that define feeders for string cells should be created below the FEEDERS declaration in your rule.

As in the case of numeric feeders, a feed to a consolidated cell results in feeding of all components of the consolidation. Because you can store strings in consolidated cells, you must pay special attention if such cells are used to feed other cells. Overuse of string feeders can result in calculation explosions and poor application performance.

For a complete discussion of TM1 rules, including the creation of feeders, please refer to *TM1 Rules*.

SKIPCHECK

You can restore sparse consolidation and improve performance by inserting a SKIPCHECK declaration at the beginning of the TM1 rule.

During consolidations, TM1 uses a sparse consolidation algorithm to skip over cells that contain zero or are empty. This algorithm speeds up consolidation calculations in cubes that are highly sparse. A sparse cube is a cube in which the number of populated cells as a percentage of total cells is low.

When consolidating data in cubes that have rules defined, TM1 turns off this sparse consolidation algorithm because one or more empty cells may in fact be calculated by a rule. (Skipping rules-calculated cells will cause consolidated totals to be incorrect). When the sparse consolidation algorithm is turned off, every cell is checked for a value during consolidation. This can slow down calculations in cubes that are very large and sparse.

```
SKIPCHECK;
```

If your rule uses a FEEDSTRINGS statement, the SKIPCHECK statement should be the second statement in your rule. If your rule does not use a FEEDSTRINGS statement, the SKIPCHECK statement should be the first statement in your rule.

When you use SKIPCHECK to restore sparse consolidation, you must also ensure that your rule includes a FEEDERS declaration and that all rules-derived cells are identified by feeder statements.

For a complete discussion of TM1 rules, including sparse consolidation and the creation of feeders, please refer to *TM1 Rules*.

TurboIntegrator Functions

TM1 TurboIntegrator lets you manipulate TM1 data and metadata when you define a process.

This is accomplished through the use of functions in the Prolog, Metadata, Data, and Epilog sub-tabs within the Advanced tab of the TurboIntegrator window. These sub-tabs include generated statements based on settings and options you select when defining a TurboIntegrator process. Any functions you create must appear after the generated statements. For details on creating processes with TurboIntegrator, see the *TM1 TurboIntegrator* documentation.

The TurboIntegrator functions in this section are sorted by category.

There is no interface to assist in the creation of TurboIntegrator functions. Enter functions by hand directly in the appropriate sub-tab within the Advanced tab. String arguments to TurboIntegrator functions must be enclosed in single quotation marks. A semi-colon (;) must be included to indicate the end of each function in the TurboIntegrator window.

In addition to these TurboIntegrator functions, you can also incorporate all standard TM1 Rules functions in a process definition, with the exception of the STET function.

Each argument to TurboIntegrator functions is limited to 256 bytes. A TurboIntegrator function can accept multiple arguments, and each argument is limited to 256 bytes.

TurboIntegrator reserved words

To prevent errors in your TurboIntegrator scripts, you should avoid creating variables with names that match any of the words listed in the following categories.

There are four categories of reserved words in TurboIntegrator:

- Rules function names - See “Rules Functions” on page 363 for a complete listing of all rules function names.
- TurboIntegrator function names - See “TurboIntegrator Functions” on page 424 for a complete listing of all TurboIntegrator function names.
- Implicit local variable names - See “TurboIntegrator Local Variables” on page 615 for a complete listing of all TurboIntegrator implicit local variables names.
- TurboIntegrator keywords - These keywords are reserved and should not be used as variables in your scripts:
  - break
  - else
  - elseif
  - end
  - endif
  - if
  - while

ASCII and Text TurboIntegrator Functions

These functions pertain to ASCII and Text.

ASCIIDelete

ASCIIDelete deletes an ASCII file.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ASCIIDelete(FileName);
```

Argument	Description
FileName	The name of the ASCII file you want to delete. If a full path is not specified, TM1 searches for the file in the server data directory.

Example

This example deletes the ASCII file named 2002Q1Results.cma from the `C:\exported_data` directory.

```
ASCIIDelete('C:\exported_data\2002Q1Results.cma');
```

ASCIIOutput

ASCIIOutput writes a comma-delimited record to an ASCII file.

This function is valid in TM1 TurboIntegrator processes only.

The ASCII file is opened when the first record is written, and is closed when the TurboIntegrator procedure (Prolog, Metadata, Data, or Epilog) containing the ASCIIIOutput function finishes processing.

Each output record generated by ASCIIOutput is limited to 8000 bytes. If an output record exceeds 8000 bytes, the record is truncated and a warning is logged in the TM1ProcessError.log file.

When ASCIIOutput encounters a String argument that pushes the output record beyond the 8000 byte limit, it ignores that argument and any further arguments. For example, if there are 10 String arguments and output for the first seven arguments total 7950 bytes while the output for the eighth argument is 51 bytes, only the output for the first seven arguments will be written to the record. If there are ten String arguments and the first argument is over 8000 bytes, no output will be written to the record.

If you use the ASCIIOutput function to write to the same file in multiple procedures (tabs) of a TurboIntegrator process, the file will be overwritten each time it is opened for a new procedure.

The ASCIIOutput function generates a minor error if an error occurs while writing the ASCII file. In addition, the function returns a value upon execution: 1 if the function successfully writes the ASCII file and 0 on failure.

Note: The error will be generated and the value returned only when ASCIIOutput is writing to a disk other than the one that the server is running on. For example, if the server is running on the C: drive and ASCIIOutput is writing to the F: drive, and the F: drive runs out of space, the error will be trapped and the server remains alive. If the server is running on the C: drive while ASCIIOutput is also writing to the C: drive, and that drive runs out of space, the server will terminate (as expected).

Note: The ability to execute the ASCIIOutput function when the data source is a cube view is determined by the **Allow Export as Text** capability assignment, which is set per user group. If a user is a member of a group which is denied the ability to export data as text, any attempt by the user to execute ASCIIOutput results in the process exiting with a permission error. The process message log indicates "`Execution was aborted. No security access for ASCIIOutput.`"

For details on how the **Allow Export as Text** capability is set, see "Capability Assignments" in *TM1 Operations*.

Note: The ASCIIOutput function places the 0x1A hexadecimal character at the end of all generated files. However, TM1 Web cannot open a Websheet that contains the 0x1A hexadecimal character.

If you use ASCIIOutput to export TM1 data to an ASCII file and then attempt to open the file in a TM1 Websheet, you will encounter the following error.

Error occurred while converting the MS Excel workbook into XML format, hexadecimal value 0x1A is an invalid character.

If you remove the 0x1A hexadecimal character from the Websheet, the file will open in TM1 Web.

Syntax

```
ASCIIOutput(FileName, String1, String2, ...Stringn);
```

Argument	Description
FileName	A full path to the ASCII file to which you want to write the record. Path must include a file extension.
String1...String*n*	A string that corresponds to each field you want to create in the ASCII file. This argument can be a string or a TurboIntegrator variable for a string.

Example

This example writes a record to the NewCube.cma ASCII file. Each field in the record corresponds to a variable assigned by TurboIntegrator to a column in your data source.

```
ASCIIOutput('NewCube.cma', V1, V2, V3, V4, V5 );
```

NumberToString

NumberToString converts a number to a string, using the decimal separator for the current user locale.

This function is valid in TM1 TurboIntegrator processes only.

In Microsoft Windows, the decimal separator is a Regional Options setting.

The output of this function is similar to the 'general' number format; it does not use thousands separators and uses the minus sign (-) to denote negative numbers.

Syntax

```
NumberToString(Value);
```

Argument	Description
Value	The real value that you want to convert to a string.

Example

```
nRET = NumberToString(1234.5);
```

NumberToStringEx

NumberToStringEx converts a number to a string, using the passed string format, decimal separator, and thousands separator.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
NumberToStringEx(Value, NumericFormat, DecimalSep, ThousandsSep);
```

Argument	Description
Value	The real value that you want to convert to a string.
FormatString	A TM1 numeric format string that defines the format for the function output. Numeric formats are described inIBM Cognos *TM1 Perspectives, TM1 Architect, and TM1 Web* documentation.
DecimalSep	The decimal separator to be used in the output string.
ThousandsSep	The thousands separator to be used in the output string.

Example

```
sRet=NUMBERTOSTRINGEX(7895.23,'#,0.#########', ',','.');
```

```
ASCIIOUTPUT('number_to_string.txt',sRet);
```

Will return in ascii file;

7.895,23

SetInputCharacterSet

SetInputCharacterSet function lets you specify the character set used in a TurboIntegrator data source.

This function is valid in TM1 TurboIntegrator processes only.

When a TurboIntegrator process reads an external file as input, it needs to know the character set in which that external file was written. If the file contains a valid byte-order-mark, TM1 functions will correctly convert the file to UTF-8 if required.

For formats lacking a valid byte-order-mark, the characters must be converted from some other encoding to UTF-8. If the proper converters are present on the machine hosting the server, the input file will be converted to the Unicode character set required by TM1.

Syntax

```
SetInputCharacterSet (CharacterSet);
```

Argument	Description
CharacterSet	The character encoding in the input file to be used by the TurboIntegrator process. If the CharacterSet argument is not a known character type, the type defaults to the system locale.

Character Encoding	System Locale
TM1CS_ISO_8859_1	ISO-8859-1 Latin-1, Western Europe
TM1CS_ISO_8859_2	ISO-8859-2 Latin-2, Central Europe
TM1CS_ISO_8859_3	ISO-8859-3 Latin-3, South Europe
TM1CS_ISO_8859_4	ISO-8859-4 Latin-4, North Europe
TM1CS_ISO_8859_5	ISO-8859-5 Latin/Cyrillic
TM1CS_ISO_8859_6	ISO-8859-6 Latin/Arabic
TM1CS_ISO_8859_7	ISO-8859-7 Latin/Greek
TM1CS_ISO_8859_8	ISO-8859-8 Latin/Hebrew
TM1CS_ISO_8859_9	ISO-8859-9 Latin-5, Turkish
TM1CS_ISO_8859_10	ISO-8859-10 Latin-6, Nordic,
TM1CS_ISO_8859_11	ISO-8859-11 Latin/Thai
TM1CS_ISO_8859_13	ISO-8859-13 Latin-7, Baltic Rim
TM1CS_ISO_8859_14	ISO-8859-14 Latin-8, Celtic
TM1CS_ISO_8859_15	ISO-8859-15 Latin-9, replaces ISO-8859-1
TM1CS_ISO_8859_16	ISO-8859-16 Latin-10, South-Eastern Europe
TM1CS_WCP1250	Microsoft Windows Central Europe

Character Encoding	**System Locale**
TM1CS_WCP1251	Windows Cyrillic
TM1CS_WCP1252	Windows Latin-1 multilingual
TM1CS_WCP1253	Windows Greek
TM1CS_WCP1254	Windows Turkish
TM1CS_WCP1255	Windows Hebrew
TM1CS_WCP1256	Windows Arabic
TM1CS_WCP1257	Windows Baltic
TM1CS_WCP1258	Windows Vietnam
TM1CS_WCP874	Windows Thai
TM1CS_WCP932	Windows Japanese
TM1CS_WCP936	Windows Simplified Chinese
TM1CS_WCP949	Windows Korean
TM1CS_WCP950	Windows Traditional Chinese
TM1CS_KOI8R	Russian and Cyrillic (KOI8-R)
TM1CS_GB18030	PRC version UNICODE
TM1CS_BIG5	Traditional Chinese
TM1CS_SHIFTJIS	JIS 0201 + JIS 0208, slightly different from CP932
TM1CS_SJIS0213	JIS 0213-2004, non-BMP required.
TM1CS_EUC_JP	EUC Japanese
TM1CS_EUC_CN	EUC Simplified Chinese
TM1CS_EUC_KR	EUC Korean
TM1CS_UTF8	UTF-8
TM1CS_UTF16	UTF-16 Little Endian
TM1CS_UTF16ESC	UNICODE notation
TM1CS_UTF32	UTF-32 Little Endian
TM1CS_OS_DEFAULT	operating system default

Character Encoding	System Locale
TM1CS_LOCALPATH	local encoding but UNICODE notation on non-native.

Example

```
SetInputCharacterSet ('TM1CS_ISO_8859_11');
```

This example specifies that the input character set for the TurboIntegrator data source is ISO-8859-11 Latin/Thai.

SetOutputCharacterSet

SetOutputCharacterSet lets you specify the character set to be used when writing to a text file using TextOutput in a TurboIntegrator proces.

This function is valid in TM1 TurboIntegrator processes only.

The function with the TextOutput function.

Used withTextOutputs.

Syntax

```
SetOutputCharacterSet( FileName, CharacterSet );
```

Argument	Description
FileName	A full path to the text file for which you want to specify a character set. The path must include a file extension. This argument should be indentical to the FileName argument for the TextOutput function.
CharacterSet	The character encoding to use when writing to the output file.

For more information on the valid values for CharacterSet, see "SetInputCharacterSet" on page 427.

SetOutputEscapeDoubleQuote

SetOutputEscapeDoubleQuote allows you to escape double quotes that appear in element names or data values when exporting a cube view to a .csv file.

This function is valid in TM1 TurboIntegrator processes only.

When SetOutputEscapeDoubleQuote is included in your TurboIntegrator script and set to 1, the exported file retains the double quote positions as they appear in your source cube view by escaping each double quote within another pair of double quotes. For example, if an element in your source view is named "Region", the element is exported as """Region""" in the .csv output file.

When SetOutputEscapeDoubleQuote is *not* included in your TurboIntegrator script or is set to 0, the exported file does not escape any double quotes that appear in your source cube.

SetOutputEscapeDoubleQuote is used in conjunction with the ASCIIOutput function, which is the function that actually writes the output file. SetOutputEscapeDoubleQuote should precede ASCIIOutput in your TurboIntegrator script, and both functions should use the same FileName parameter value.

Syntax

```
SetOutputEscapeDoubleQuote(FileName, Num);
```

Argument	Description
FileName	A full path to the file to which you want to write the cube view. Path must include a file extension.
Num	A flag that determines if double quotes are escaped in the output file. 1 indicates that double quotes will be escaped in the output file. 0 indicates that double quotes will not be escaped in the output file.

Example

```
SetOutputEscapeDoubleQuote('C:\temp\cube1.csv', 1);
```

This example escapes any double quotes encountered in the source cube view when writing output to the C:\temp\cube1.csv file.

StringToNumber

StringToNumber converts a string to a number, using the decimal separator for the current user locale. If the input string is an invalid number string, the value returned will be an invalid floating point value. In Microsoft Windows, the decimal separator is a Regional Options setting.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
StringToNumber(String);
```

Argument	Description
String	The string you want to convert to a number.

Example

```
nRET = StringToNumber('123.45');
```

StringToNumberEx

StringToNumberEx converts a string to a number, using the passed decimal separator and thousands separator. If the input string is an invalid number string, the value returned will be an invalid floating point value.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
StringToNumberEx(String, DecimalSep, ThousandsSep);
```

Argument	Description
String	The string that you want to convert to a number.
DecimalSep	The decimal separator to be used in the output number.
ThousandsSep	The thousands separator to be used in the output number.

Example

```
nRET = StringToNumberEx('12453.45', ' . ', ' , ');
```

TextOutput

TextOutput writes a comma-delimited record to a text file.

This function is valid in TM1 TurboIntegrator processes only.

By default TextOutput writes characters in the locale character set of the server machine. To create a file in a different character set, call the function SetOutputCharacterSet before calling TextOutput.

The text file is opened when the first record is written, and is closed when the TurboIntegrator procedure (Prolog, Metadata, Data, or Epilog) containing the TextOutput function finishes processing.

If you use the TextOutput function to write to the same file in multiple procedures (tabs) of a TurboIntegrator process, the file will be overwritten each time it is opened for a new procedure.

Each output record generated by TextOutput is limited to 8000 bytes. If an output record exceeds 8000 bytes, the record is truncated and a warning is logged in the TM1ProcessError.log file.

When TextOutput encounters a String argument that pushes the output record beyond the 8000 byte limit, it ignores that argument and any further arguments. For example, if there are 10 String arguments and output for the first seven arguments total 7950 bytes while the output for the eighth argument is 51 bytes, only the output for the first seven arguments will be written to the record. If there are ten String arguments and the first argument is over 8000 bytes, no output will be written to the record.

The TextOutput function generates a minor error if an error occurs while writing the text file. In addition, the function returns a value upon execution: 1 if the function successfully writes the text file and 0 on failure.

The error will be generated and the value returned only when TextOutput is writing to a disk other than the one that the server is running on. For example, if the server is running on the C: drive and TextOutput is writing to the F: drive, and the F: drive runs out of space, the error will be trapped and the server remains alive. If the server is running on the C: drive while TextOutput is also writing to the C: drive, and that drive runs out of space, the server will terminate (as expected).

Note: The ability to execute the TextOutput function when the data source is a cube view is determined by the **Allow Export as Text** capability assignment, which is set per user group. If a user is a member of a group which is denied the ability to export data as text, any attempt by the user to execute TextOutput results in the process exiting with a permission error. The process message log indicates "`Execution was aborted. No security access for TextOutput.`"

For details on how the **Allow Export as Text** capability is set, see "Capability Assignments" in the *IBM Cognos TM1 Operations* documentation.

Syntax

```
TextOutput(FileName, String1, String2, ...Stringn);
```

Argument	Description
FileName	A full path to the text file to which you want to write the record. Path must include a file extension.
String1...String*n*	A string that corresponds to each field you want to create in the text file. This argument can be a string or a TurboIntegrator variable for a string.

Example

```
TextOutput('NewCube.cma', V1, V2, V3, V4, V5 );
```

This example writes a record to the NewCube.cma file. Each field in the record corresponds to a variable assigned by TurboIntegrator to a column in your data source.

Attribute Manipulation TurboIntegrator Functions

These functions facilitate the manipulation of attributes.

ATTRNL

ATTRNL returns a numeric attribute for a specified element of a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ATTRNL(DimName, ElName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
ElName	An element of the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Engine Size attribute of the L Series 1.8L Sedan element in the Model dimension for the French locale.

```
ATTRNL('Model', 'L Series 1.8L Sedan', 'Engine Size', 'fr');
```

ATTRSL

AttrSL returns a string attribute for a specified element of a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AttrSL(DimName, ElName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
ElName	An element of the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Currency attribute of the 10100 element in the Plan_Business_Unit dimension for the French locale.

```
AttrSL('Plan_Business_Unit', '10100', 'Currency', 'fr');
```

AttrDelete

AttrDelete deletes an element attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AttrDelete(DimName, AttrName);
```

Argument	Description
DimName	The dimension for which you want to delete an element attribute.
AttrName	The name of the attribute you want to delete.

Example

This example deletes the InteriorColor element attribute for the Model dimension.

```
AttrDelete('Model', 'InteriorColor');
```

AttrInsert

AttrInsert creates a new element attribute for a dimension. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AttrInsert(DimName, PrevAttr, AttrName, Type);
```

Argument	Description
DimName	The dimension for which you want to create an element attribute.
PrevAttr	The attribute that precedes the attribute you are creating.
AttrName	The name you want to assign to the new attribute.
Type	The type of attribute. There are three possible values for the Type argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the InteriorColor string attribute for the Model dimension. This attribute is inserted after the Transmission attribute.

```
AttrInsert('Model', 'Transmission', 'InteriorColor','S');
```

AttrPutN

AttrPutN assigns a value to a numeric element attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AttrPutN( Value, DimName, ElName, AttrName, [LangLocaleCode] );
```

Argument	Description
Value	The numeric value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.
ElName	The element for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 2257993 to the ProdCode attribute of the S Series 1.8L Sedan in the Model dimension.

```
AttrPutN(2257993, 'Model', ' S Series 1.8L Sedan ','ProdCode');
```

AttrPutS

AttrPutS assigns a value to a string element attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AttrPutS(Value, DimName, ElName, AttrName, [LangLocaleCode] );
```

Argument	Description
Value	The value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.
ElName	The element for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the Value applies. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string Beige to the InteriorColor attribute of the S Series 1.8L Sedan in the Model dimension.

```
AttrPutS('Beige', 'Model', 'S Series 1.8L Sedan', 'InteriorColor');
```

ChoreAttrDelete

ChoreAttrDelete deletes a chore attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrDelete(AttrName);
```

Argument	Description
AttrName	The name of the chore attribute you want to delete.

Example

This example deletes the Description attribute for chores on your TM1 server.

```
ChoreAttrDelete('Description');
```

ChoreAttrInsert

ChoreAttrInsert creates a new attribute for chores on your TM1 server. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

⚠ **Attention:** If you update an existing chore attribute, you must first delete the existing attribute using the function ChoreAttrDelete. You can then use ChoreAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
ChoreAttrInsert( PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for chores, leave this argument empty.
NewAttrName	The name you want to assign to the new chore attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for chores. This attribute is inserted after the Owner attribute.

```
ChoreAttrInsert('Owner', 'Description', 'S');
```

ChoreAttrN

ChoreAttrN returns a numeric attribute for a specified chore.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrN(ChoreName, AttrName);
```

Argument	Description
ChoreName	A valid chore name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the chore.

Example

In this example, the function returns the numeric value of the Division_Code attribute of the Import chore.

```
ChoreAttrN('Import', 'Division_Code');
```

ChoreAttrNL

ChoreAttrNL returns an attribute's numeric value for a specified chore with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrNL(ChoreName, AttrName, [LangLocaleCode]);
```

Argument	Description
ChoreName	A valid chore name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the chore.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Division_Code attribute of the Import chore for the French locale.

```
ChoreAttrNL('Import', 'Division_Code', 'fr');
```

ChoreAttrPutN

ChoreAttrPutN assigns a value to a numeric chore attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrPutN(NumericValue, ChoreName, AttrName, [LangLocaleCode] );
```

Argument	Description
NumericValue	The value you want to assign to a chore attribute.
ChoreName	The chore for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 7161994 to the Division_Code attribute of the Import chore for the French language locale code.

```
ChoreAttrPutN(7161994, 'Import', 'Division_Code','fr');
```

ChoreAttrPutS

ChoreAttrPutS assigns a value to a string chore attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrPutS(String, ChoreName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a chore attribute.
ChoreName	The chore for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Ricci to the Owner attribute of the Import chore, for the French language locale code.

```
ChoreAttrPutS('Ricci', 'Import', 'Owner', 'fr');
```

ChoreAttrS

ChoreAttrS returns a string attribute for a specified chore.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrS(ChoreName, AttrName);
```

Argument	Description
ChoreName	A valid chore name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the chore.

Example

In this example, the function returns the string value of the Owner attribute of the Exchange_Rate_Updates chore.

```
ChoreAttrS('Exchange_Rate_Updates', 'Owner');
```

ChoreAttrSL

ChoreAttrSL returns a string attribute value for a specified chore with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreAttrSL(ChoreName, AttrName, [LangLocaleCode]);
```

Argument	Description
ChoreName	A valid chore name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the chore.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Owner attribute of the Depreciate_Inventory chore for the French locale.

```
ChoreAttrSL('Depreciate_Inventory', 'Owner', 'fr');
```

CreateHierarchyByAttribute

CreateHierarchyByAttribute creates a simple 3-level hierarchy from a single attribute.

The new hierarchy consists of a single high-level root element, a middle-level of consolidations representing existing attribute values, and a lower-level of dimension leaves that include the associated attribute value.

This function is valid in TM1 TurboIntegrator processes only.

Note: This function creates a hierarchy from the current set of attribute values, but the system does not automatically keep the hierarchy in-sync with the attribute data as it changes. Modelers must regenerate the hierarchy as needed.

Syntax

```
CreateHierarchyByAttribute(DimName, AttrName [, emptyParent [, rootName ] ] );
```

Argument	Description
DimName	The name of the dimension that contains the attribute. A hierarchy of the same name as the dimension will be created.

Argument	Description
AttrName	The name of the attribute to create the hierarchy from.
emptyParent	Specifies the name of a consolidation to create, which collects dimension leaves that don't have an attribute value. If passed as an empty string, the function does not create a consolidation.
rootName	Overrides the root element name which by default is named after the attribute.

Example

```
CreateHierarchyByAttribute ('Country', 'City');
```

This example creates a hierarchy from the City attribute in the Country dimension.

CubeAttrDelete

CubeAttrDelete deletes a cube attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeAttrDelete(AttrName);
```

Argument	Description
AttrName	The name of the cube attribute you want to delete.

Example

This example deletes the Description attribute for cubes on your TM1 server.

```
CubeAttrDelete('Description');
```

CubeAttrInsert

CubeAttrInsert creates a new attribute for cubes on your TM1 server. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

⚠ **Attention:** If you update an existing cube attribute, you must first delete the existing attribute using the function CubeAttrDelete. You can then use CubeAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
CubeAttrInsert( PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for cubes, leave this argument empty.
NewAttrName	The name you want to assign to the new cube attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for cubes. This attribute is inserted after the Owner attribute.

```
CubeAttrInsert('Owner', 'Description', 'S');
```

CubeAttrPutN

CubeAttrPutN assigns a value to a numeric cube attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeAttrPutN(NumericValue, CubeName, AttrName, [LangLocaleCode] );
```

Argument	Description
NumericValue	The value you want to assign to a cube attribute.
CubeName	The cube for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 07161994 to the AccountingCode attribute of the Sales cube for the French language locale code.

```
CubeAttrPutN(07161994, 'Sales', 'AccountingCode','fr');
```

CubeAttrPutS

CubeAttrPutS assigns a value to a string cube attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeAttrPutS(String, CubeName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a cube attribute.
CubeName	The cube for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Prototype to the Description attribute of the Sales cube for the French language locale code.

```
CubeAttrPutS('Prototype', 'Sales', 'Description','fr');
```

CubeATTRNL

CubeATTRNL returns a numeric attribute value for a specified cube with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes.

Syntax

```
CubeATTRNL(CubeName, AttrName, [LangLocaleCode]);
```

Argument	Description
CubeName	A valid cube name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the cube.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Product cube for the French locale.

```
CubeATTRNL('Product', 'Accounting_Code', 'fr');
```

CubeATTRSL

CubeATTRSL returns a string attribute value for a specified cube with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes.

Syntax

```
CubeATTRSL(CubeName, AttrName, [LangLocaleCode]);
```

Argument	Description
CubeName	A valid cube name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the cube.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Owner attribute of the Product cube for the French locale.

```
CubeATTRSL('Product', 'Owner', 'fr');
```

DimensionAttrDelete

DimensionAttrDelete deletes a dimension attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionAttrDelete(AttrName);
```

Argument	Description
AttrName	The name of the dimension attribute you want to delete.

Example

This example deletes the Description attribute for dimensions on your TM1 server.

```
DimensionAttrDelete('Description');
```

DimensionAttrInsert

DimensionAttrInsert creates a new attribute for dimensions on your TM1 server. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

⚠ **Attention:** If you update an existing dimension attribute, you must first delete the existing attribute using the function DimensionAttrDelete. You can then use DimensionAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
DimensionAttrInsert( PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for dimensions, leave this argument empty.
NewAttrName	The name you want to assign to the new dimension attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for dimensions. Because there is no PrevAttrName parameter, this attribute is inserted as the first attribute for dimensions on your TM1 server.

```
DimensionAttrInsert('', 'Description', 'S');
```

DimensionAttrPutN

DimensionAttrPutN assigns a value to a numeric dimension attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionAttrPutN(NumericValue, DimensionName, AttrName, [LocalLangCode] );
```

Argument	Description
NumericValue	The value you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.

Argument	Description
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 07161994 to the AccountingCode attribute of the Models dimension for the French language locale code.

```
DimensionAttrPutN(07161994, 'Models', 'AccountingCode','fr');
```

DimensionAttrPutS

DimensionAttrPutS assigns a value to a string dimension attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionAttrPutS(String, DimensionName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Prototype to the Description attribute of the Model dimension for the French language locale code.

```
DimensionAttrPutS('Prototype', 'Model', 'Description','fr');
```

DimensionATTRNL

DimensionATTRNL returns a numeric attribute value for a specified dimension with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionATTRNL(DimName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Plan_Business_Unit dimension for the French locale.

```
DimensionATTRNL('Plan_Business_Unit', 'Accounting_Code', 'fr');
```

DimensionATTRSL

DimensionATTRSL returns a string attribute value for a specified dimension with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionATTRSL(DimName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Manager attribute of the Plan_Business_Unit dimension for the French locale.

```
DimensionATTRSL('Plan_Business_Unit', 'Manager', 'fr');
```

ElementATTRNL

ElementATTRNL returns a numeric attribute for a specified element of a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementATTRNL(DimName, HierName, ElName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.

Argument	Description
HierName	The name of the hierarchy within the dimension.
ElName	An element of the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Engine Size attribute of the L Series 1.8L Sedan element in the Model dimension for the French locale. This example applies to the 2015 hierarchy.

```
ATTRNL('Model', '2015', 'L Series 1.8L Sedan', 'Engine Size', 'fr');
```

ElementATTRSL

ElementATTRSL returns a string attribute for a specified element of a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementATTRSL(DimName, HierName, ElName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of the hierarchy within the dimension.

Argument	Description
ElName	An element of the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Currency attribute of the 10100 element in the Plan_Business_Unit dimension for the French locale.

```
ElementATTRSL('Plan_Business_Unit', '10100', 'Currency', 'fr');
```

ElementAttrPutN

ElementAttrPutN assigns a value to a numeric element attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementAttrPutN( Value, DimName, HierName, ElName, AttrName, [LangLocaleCode] );
```

Argument	Description
Value	The numeric value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.

Argument	Description
HierName	The name of the hierarchy within the dimension.
ElName	The element for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 2257993 to the ProdCode attribute of the S Series 1.8L Sedan in the Automobile hierarchy of the Model dimension.

```
ElementAttrPutN(2257993, 'Model', 'Automobile', ' S Series 1.8L Sedan ','ProdCode');
```

ElementAttrPutS

ElementAttrPutS assigns a value to a string element attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementAttrPutS(Value, DimName, HierName, ElName, AttrName, [LangLocaleCode] );
```

Argument	Description
Value	The value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.
HierName	The name of the hierarchy within the dimension.
ElName	The element for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the Value applies. Valid LangLocaleCode values correspond to the international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string Beige to the InteriorColor attribute of the S Series 1.8L Sedan in the Automobile hierarchy of the Model dimension.

```
ElementAttrPutS('Beige', 'Model', 'Automobile', 'S Series 1.8L Sedan', 'InteriorColor');
```

ElementAttrInsert

ElementAttrInsert creates a new element attribute for a dimension. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementAttrInsert(DimName, HierName, PrevAttr, AttrName, Type);
```

Argument	Description
DimName	The dimension for which you want to create an element attribute.
HierName	The name of the hierarchy within the dimension.
PrevAttr	The attribute that precedes the attribute you are creating.
AttrName	The name you want to assign to the new attribute.
Type	The type of attribute. There are three possible values for the Type argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the InteriorColor string attribute in the Automobile hierarchy in the Model dimension. This attribute is inserted after the Transmission attribute.

```
ElementAttrInsert('Model', 'Automobile', 'Transmission', 'InteriorColor','S');
```

ElementAttrDelete

ElementAttrDelete deletes an element attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementAttrDelete(DimName, HierName, AttrName);
```

Argument	Description
DimName	The dimension for which you want to delete an element attribute.
HierName	The name of the hierarchy within the dimension.
AttrName	The name of the attribute you want to delete.

Example

This example deletes the InteriorColor element attribute from the Autombile hierarchy in the Model dimension.

```
ElementAttrDelete('Model', 'Automobile', 'InteriorColor');
```

HierarchyAttrPutN

HierarchyAttrPutN assigns a value to a numeric attribute in a specified hierarchy within a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyAttrPutN(NumericValue, DimensionName, HierName, AttrName, [LocalLangCode] );
```

Argument	Description
NumericValue	The value you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.
HierName	The name of the hierarchy within the dimension.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 07161994 to the AccountingCode attribute of the Models dimension for the French language locale code. This change is applied to the Receivables hierarchy in the Models dimension.

```
HierarchyAttrPutN(07161994, 'Models', 'Receivables', 'AccountingCode','fr');
```

HierarchyAttrPutS

HierarchyAttrPutS assigns a value to a string attribute in a specified hierarchy within a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyAttrPutS(String, DimensionName, HierName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.
HierName	The name of the hierarchy within the dimension.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Prototype to the Description attribute of the Model dimension for the French language locale code. This change is applied to the Receivables hierarchy in the Model dimension.

```
HierarchyAttrPutS('Prototype', 'Model', 'Receivables', 'Description','fr');
```

HierarchyATTRN

HierarchyATTRN returns a numeric attribute for a specified hierarchy within a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyATTRN(DimName, HierName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of the hierarchy within the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Plan_Business_Unit dimension. This example applies to the Equipment hierarchy.

```
HierarchyATTRN('Plan_Business_Unit', 'Equipment', 'Accounting_Code');
```

HierarchyATTRS

HierarchyATTRS returns a string attribute for a specified hierarchy within a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyATTRS(DimName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of the hierarchy within the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the string value of the Manager attribute of the Plan_Business_Unit dimension. This example applies to the Equipment hierarchy.

```
HierarchyATTRS('Plan_Business_Unit', 'Equipment', 'Manager');
```

HierarchyATTRNL

HierarchyATTRNL returns a numeric attribute value for a specified hierarchy within a dimension with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyATTRNL(DimName, HierName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.

Argument	Description
HierName	The name of the hierarchy within the dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Plan_Business_Unit dimension for the French locale. This function applies to the Equipment hierarchy.

```
HierarchyATTRNL('Plan_Business_Unit', 'Equipment', 'Accounting_Code', 'fr');
```

HierarchyATTRSL

HierarchyATTRSL returns a string attribute value for a specified hierarchy within a dimension with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyATTRSL(DimName, HierName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of the hierarchy within the dimension.

Argument	Description
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Manager attribute of the Plan_Business_Unit dimension for the French locale. This function applies to the Equipment hierarchy.

```
HierarchyATTRSL('Plan_Business_Unit', 'Equipment', 'Manager', 'fr');
```

HierarchySubsetATTRS

HierarchySubsetATTRS returns a string attribute for a specified subset associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetATTRS(DimName, HierName, SubName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.

Argument	Description
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the string value of the Manager attribute of the Sales subset from Europe hierarchy in the Plan_Business_Unit dimension.

```
HierarchySubsetATTRS('Plan_Business_Unit', 'Europe', 'Sales', 'Manager');
```

HierarchySubsetATTRN

HierarchySubsetATTRN returns a numeric attribute for a specified subset associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetATTRN(DimName, HierName, SubName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Sales subset from the Europe hierarchy in the Plan_Business_Unit dimension.

```
HierarchySubsetATTRN('Plan_Business_Unit', 'Europe', 'Sales', 'Accounting_Code');
```

HierarchySubsetATTRSL

HierarchySubsetATTRSL returns an attribute's string value for a specified subset (and locale) associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetATTRSL(DimName, HierName, SubName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.

Argument	Description
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Manager attribute of the Sales subset (from the Europe hierarchy) for the French locale.

```
HierarchySubsetATTRSL('Plan_Business_Unit', 'Europe', 'Sales', 'Manager', 'fr');
```

HierarchySubsetATTRNL

HierarchySubsetATTRNL returns an attribute's numeric value for a specified subset (and locale) associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetATTRNL(DimName, HierName, SubName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.

Argument	Description
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Sales subset (from the Europe hierarchy) for the French locale.

```
HierarchySubsetATTRNL('Plan_Business_Unit', 'Europe', 'Sales', 'Accounting_Code', 'fr');
```

HierarchySubsetAttrPutS

HierarchySubsetAttrPutS assigns a string value to an attribute for a specified subset associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetAttrPutS(String, DimName, HierName, SubName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a dimension attribute.
DimName	The dimension for which you want to assign an attribute value.
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Prototype to the Description attribute of the Z subset (from the 2016 hierarchy in the Model dimension) for the French language locale code.

```
HierarchySubsetAttrPutS('Prototype', 'Model', '2016', 'Z', 'Description','fr');
```

HierarchySubsetAttrPutN

HierarchySubsetAttrPutN assigns a numeric value to an attribute for a specified subset associated with a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetAttrPutN(NumericValue, DimName, HierName, SubName, AttrName, [LocalLangCode] );
```

Argument	Description
NumericValue	The value you want to assign to a dimension attribute.
DimName	The dimension for which you want to assign an attribute value.
HierName	The name of a hierarchy in a dimension.
SubName	The name of a subset in a dimension.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 07161994 to the AccountingCode attribute of the Z subset (from the 2016 hierarchy in the Models dimension) for the French language locale code.

```
HierarchySubsetAttrPutN(07161994, 'Models', '2016', 'Z', 'AccountingCode','fr');
```

HierarchySubsetAttrInsert

HierarchySubsetAttrInsert creates a new attribute for subsets on your TM1 server. The function creates a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

⚠ **Attention:** If you update an existing subset attribute, you must first delete the existing attribute using the function HierarchySubsetAttrDelete. You can then use HierarchySubsetAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

```
HierarchySubsetAttrInsert( Dimension, Hierarchy, PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
Dimension	The name of the dimension whose subsets are being updated.
Hierarchy	The name of a hierarchy in a dimension.
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for subsets, leave this argument empty.
NewAttrName	The name you want to assign to the new subset attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for subsets in the Z hierarchy of the Model dimension. Because there is no PrevAttrName parameter, this attribute is inserted as the first attribute for subsets on your TM1 server.

```
HierarchySubsetAttrInsert('Model', 'Z', '', 'Description', 'S');
```

HierarchySubsetAttrDelete

HierarchySubsetAttrDelete deletes a subset attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetAttrDelete(Dimension, Hierarchy, AttrName);
```

Argument	Description
Dimension	The name of the dimension whose subset attribute is being deleted.
Hierarchy	The name of a hierarchy in a dimension.
AttrName	The name of the dimension attribute you want to delete.

Example

This example deletes the Description attribute for subsets from the Z hierarchy in the Model dimension.

```
HierarchySubsetAttrDelete('Model', 'Z, 'Description');
```

ProcessAttrDelete

ProcessAttrDelete deletes a process attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrDelete(AttrName);
```

Argument	Description
AttrName	The name of the process attribute you want to delete.

Example

This example deletes the Description attribute for processes on your TM1 server.

```
ProcessAttrDelete('Description');
```

ProcessAttrInsert

ProcessAttrInsert creates a new attribute for processes on your TM1 server. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

⚠ **Attention:** If you update an existing process attribute, you must first delete the existing attribute using the function ProcessAttrDelete. You can then use ProcessAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
ProcessAttrInsert( PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for processes, leave this argument empty.
NewAttrName	The name you want to assign to the new process attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for processes. This attribute is inserted after the Owner attribute.

```
ProcessAttrInsert('Owner', 'Description', 'S');
```

ProcessAttrN

ProcessAttrN returns a numeric attribute for a specified process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrN(ProcessName, AttrName);
```

Argument	Description
ProcessName	A valid process name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the process.

Example

In this example, the function returns the numeric value of the Store_Code attribute of the Daily_Sales process.

```
ProcessAttrN('Daily_Sales', 'Store_Code');
```

ProcessAttrNL

ProcessAttrNL returns an attribute's numeric value for a specified process with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrNL(ProcessName, AttrName, [LangLocaleCode]);
```

Argument	Description
ProcessName	A valid process name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the process.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Store_Code attribute of the Daily_Sales process for the French locale.

```
ProcessAttrNL('Daily_Sales', 'Store_Code', 'fr');
```

ProcessAttrPutN

ProcessAttrPutN assigns a value to a numeric process attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrPutN(NumericValue, CubeName, AttrName, [LangLocaleCode] );
```

Argument	Description
NumericValue	The value you want to assign to a process attribute.
ProcessName	The process for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 8051997 to the Store_Code attribute of the Daily_Sales process for the French language locale code.

```
ProcessAttrPutN(8051997, 'Daily_Sales', 'Store_Code','fr');
```

ProcessAttrPutS

ProcessAttrPutS assigns a value to a string process attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrPutS(String, ProcessName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a process attribute.
ProcessName	The process for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Ricci to the Owner attribute of the Import_Transactional process, for the French language locale code.

```
ProcessAttrPutS('Ricci', 'Import_transactional', 'Owner', 'fr');
```

ProcessAttrS

ProcessAttrS returns a string attribute for a specified process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrS(ProcessName, AttrName);
```

Argument	Description
ProcessName	A valid process name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the process.

Example

In this example, the function returns the string value of the Owner attribute of the Refresh_Cubes process.

```
ProcessAttrS('Refresh_Cubes', 'Owner');
```

ProcessAttrSL

ProcessAttrSL returns a string attribute value for a specified process with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessAttrSL(ProcessName, AttrName, [LangLocaleCode]);
```

Argument	Description
ProcessName	A valid process name.

Argument	Description
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the process.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Owner attribute of the Exchange_Rate_Update process for the French-Canada locale.

```
ProcessAttrSL('Exchange_Rate_Update', 'Owner', 'fr-CA');
```

SubsetATTRS

SubsetATTRS returns a string attribute for a specified subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetATTRS(DimName, SubName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
SubName	The name of a subset in a dimension.

Argument	Description
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the string value of the Manager attribute of the Sales subset from the Plan_Business_Unit dimension.

```
SubsetATTRS('Plan_Business_Unit', 'Sales', 'Manager');
```

SubsetATTRN

SubsetATTRN returns a numeric attribute for a specified subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetATTRN(DimName, SubName, AttrName);
```

Argument	Description
DimName	A valid dimension name.
SubName	The name of a subset in a dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Sales subset from the Plan_Business_Unit dimension.

```
SubsetATTRN('Plan_Business_Unit', 'Sales', 'Accounting_Code');
```

SubsetATTRSL

SubsetATTRSL returns an attribute's string value for a specified subset with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetATTRSL(DimName, SubName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
SubName	The name of a subset in a dimension.

Argument	Description
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Manager attribute of the Sales subset for the French locale.

```
SubsetATTRSL('Plan_Business_Unit', 'Sales', 'Manager', 'fr');
```

SubsetATTRNL

SubsetATTRNL returns an attribute's numeric value for a specified subset with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetATTRNL(DimName, SubName, AttrName, [LangLocaleCode]);
```

Argument	Description
DimName	A valid dimension name.
SubName	The name of a subset in a dimension.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the dimension.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value of the Accounting_Code attribute of the Sales subset for the French locale.

```
SubsetATTRNL('Plan_Business_Unit', 'Sales', 'Accounting_Code', 'fr');
```

SubsetAttrPutS

SubsetAttrPutS assigns a string value to an attribute for a specified subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetAttrPutS(String, DimensionName, SubName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.
SubName	The name of a subset in a dimension.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Prototype to the Description attribute of the Z subset (from the Model dimension) for the French language locale code.

```
SubsetAttrPutS('Prototype', 'Model', 'Z', 'Description','fr');
```

SubsetAttrPutN

SubsetAttrPutN assigns a numeric value to an attribute for a specified subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetAttrPutN(NumericValue, DimensionName, SubName, AttrName, [LocalLangCode] );
```

Argument	Description
NumericValue	The value you want to assign to a dimension attribute.
DimensionName	The dimension for which you want to assign an attribute value.
SubName	The name of a subset in a dimension.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 07161994 to the AccountingCode attribute of the Z subset (from the Models dimension) for the French language locale code.

```
SubsetAttrPutN(07161994, 'Models', 'Z', 'AccountingCode','fr');
```

SubsetAttrInsert

SubsetAttrInsert creates a new attribute for subsets on your TM1 server. The function creates a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

Attention: If you update an existing subset attribute, you must first delete the existing attribute using the function SubsetAttrDelete. You can then use SubsetAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
SubsetAttrInsert( Dimension, PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
Dimension	The name of the dimension whose subsets are being updated.
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for subsets, leave this argument empty.
NewAttrName	The name you want to assign to the new subset attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for subsets in the Model dimension. Because there is no PrevAttrName parameter, this attribute is inserted as the first attribute for subsets on your TM1 server.

```
SubsetAttrInsert('Model', '', 'Description', 'S');
```

SubsetAttrDelete

SubsetAttrDelete deletes a subset attribute from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetAttrDelete(Dimension, AttrName);
```

Argument	Description
Dimension	The name of the dimension whose subset attribute is being deleted.
AttrName	The name of the dimension attribute you want to delete.

Example

This example deletes the Description attribute for subsets in the Model dimension.

```
SubsetAttrDelete('Model', 'Description');
```

ViewAttrDelete

ViewAttrDelete deletes a view attribute for a specific cube from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrDelete(CubeName, AttrName);
```

Argument	Description
CubeName	The name of the cube whose view attribute is being deleted.
AttrName	The name of the view attribute you want to delete.

Example

This example deletes the Description attribute for views of the Sales cube on your TM1 server.

```
ViewAttrDelete('Sales', 'Description');
```

ViewAttrInsert

ViewAttrInsert creates a new attribute for views of a specific cube on your TM1 server. The function can create a string, numeric, or alias attribute.

This function is valid in TM1 TurboIntegrator processes only.

⚠ **Attention:** If you update an existing view attribute, you must first delete the existing attribute using the function ViewAttrDelete. You can then use ViewAttrInsert to recreate the attribute with your desired changes.

Important: If you attempt to update an existing attribute without first deleting it, the insert fails without a warning or error. The existing attribute remains unchanged; it is neither updated nor overwritten.

Syntax

```
ViewAttrInsert( CubeName, PrevAttrName, NewAttrName, AttrType);
```

Argument	Description
CubeName	The parent cube for which you want to insert a view attribute.

Argument	Description
PrevAttrName	The attribute that precedes the attribute you are creating. If there is no previous attribute or you want the new attribute to be the first attribute for views, leave this argument empty.
NewAttrName	The name you want to assign to the new view attribute.
AttrType	The type of attribute. There are three possible values for the AttrType argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

This example creates the Description string attribute for views of the Sales cube. This attribute is inserted after the Owner attribute.

```
ViewAttrInsert('Sales', 'Owner', 'Description', 'S');
```

ViewAttrN

ViewAttrN returns a numeric attribute for a specified view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrN(CubeName, ViewName, AttrName);
```

Argument	Description
CubeName	A valid cube name.
ViewName	A valid view name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the view.

Example

In this example, the function returns the numeric value for the Category_Code attribute of the Product view of the Sales cube.

```
ViewAattrN('Sales', 'Product', 'Category_Code');
```

ViewAttrNL

ViewAttrNL returns an attribute's numeric value for a specified view with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrNL(CubeName, ViewName, AttrName, [LangLocaleCode]);
```

Argument	Description
CubeName	The parent cube for the view whose attribute value you want to retrieve.
ViewName	A valid view name.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the view.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the numeric value for the Category_Code attribute of the Product view of the Sales cube, for the French locale.

```
ViewAttrNL('Sales', 'Product', 'Category_Code', 'fr');
```

ViewAttrPutN

ViewAttrPutN assigns a value to a numeric view attribute.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrPutN(NumericValue, CubeName, ViewName, AttrName, [LangLocaleCode] );
```

Argument	Description
NumericValue	The value you want to assign to a view attribute.
CubeName	The parent cube of the view for which you want to assign an attribute value.
ViewName	The view for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the value 8222001 to the Category_Code attribute of the Product view of the Sales cube, for the French language locale code.

```
ViewAttrPutN(8222001, 'Sales', 'Product', 'Category_Code','fr');
```

ViewAttrPutS

ViewAttrPutS assigns a string value to an attribute for a specified view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrPutS(String, CubeName, ViewName, AttrName, [LangLocaleCode] );
```

Argument	Description
String	The string you want to assign to a view attribute.
CubeName	The cube parent of the view for which you want to assign an attribute value.
ViewName	The name of the view for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Argument	Description
LangLocaleCode	This optional parameter specifies the language locale code to which the NumericValue applies. Valid LangLocaleCode values correspond to the ISO 639-1 international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the base attribute value is updated.

Example

This example assigns the string value Rocheford to the Owner attribute of the Individual_Stores view of the Sales cube, for the French language locale code.

```
ViewAttrPutS('Rocheford', 'Sales', 'Individual_Stores', 'Owner','fr');
```

ViewAttrS

ViewAttrS returns a string attribute for a specified view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrS(CubeName, ViewName, AttrName);
```

Argument	Description
CubeName	The parent cube of the view for which you want to return an attribute value.
ViewName	The view for which you want to return an attribute value.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the view.

Example

```
ViewAttrS('Plan_Business_Unit', 'Sales', 'Manager');
```

In this example, the function returns the string value of the Manager attribute of the Sales view of the Plan_Business_Unit cube.

ViewAttrSL

ViewAttrSL returns an attribute's string value for a specified view with respect to a given locale.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewAttrSL(CubeName, ViewName, AttrName, [LangLocaleCode]);
```

Argument	Description
CubeName	The parent cube of the view for which you want to return an attribute value.
ViewName	The view for which you want to return an attribute value.
AttrName	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the view.
LangLocaleCode	This optional parameter specifies the language locale code for which you want to return a value. Valid LangLocaleCode values correspond to the international language codes listed in the }Cultures control dimension. When the LangLocaleCode is not specified or is omitted, the user's current locale is used as the LangLocaleCode argument. If an attribute value does not exist for the LangLocaleCode, the value for an associated parent LangLocaleCode is returned. If an attribute value does not exist for an associated parent LangLocaleCode, the base attribute value is returned. For example if the LangLocaleCode is `fr-CA`, the function returns the attribute value for the `fr-CA` (French-Canada) locale if available. If the attribute value for `fr-CA` is not available, the function attempts to return the attribute value for the parent `fr` (French) locale. If the attribute value for `fr` is not available, the base attribute value is returned

Example

In this example, the function returns the string value of the Manager attribute of the Sales view of the Plan_Business_Unit cube, for the French-Canada locale.

```
ViewAttrSL('Plan_Business_Unit', 'Sales', 'Manager', 'fr-CA');
```

Chore Management TurboIntegrator Functions

These functions pertain to managing chores.

ChoreError

ChoreError causes the immediate termination of a chore. It can be called from any process within a chore. The ChoreError TurboIntegrator function causes an immediate termination of a single chore. Chores terminated with this function are flagged with an error status.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreError;
```

Arguments

None.

ChoreQuit

ChoreQuit causes the immediate termination of a chore. It can be called from any process within a chore. The current chore is terminated with an error status, and a message is written to the server log file indicating that ChoreQuit was called to terminate the chore.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreQuit;
```

Arguments

None.

ChoreRollback

ChoreRollback initiates a chore rollback. When used inside a TurboIntegrator process, this function throws out all pending edits and cancels further processing. An error message appears in the tm1server.log and tm1processorerrorXXX.log files.

When used in a single-commit mode chore, **ChoreRollback** throws out all pending edits from all previous processes and chore execution stops with an error code. When used in a multi-commit mode chore, **ChoreRollback** throws out all pending edits from the current processes and chore execution stops with an error code. Changes that have already been committed cannot be rolled back.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ChoreRollback;
```

Arguments

None.

SetChoreVerboseMessages

SetChoreVerboseMessages is used to turn on (or off) more verbose reporting of messages to the Tm1s.log file. You can use this function to debug chores in which several processes call each other with the ExecuteProcess function.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetChoreVerboseMessages(Flag);
```

Passing a zero value turns off the output of these messages, passing a non-zero value enables the output of more verbose messages. By default this flag is off.

Use this function to turn on (or off) more verbose reporting of messages to the Tm1s.log file. This function is best used as an aid to debugging chores in which several processes call one another through use of the ExecuteProcess function.

Passing a zero value turns off the output of these messages, passing a non-zero value enables the output of more verbose messages. By default this flag is off.

Argument	Description
Flag	Set to a non-zero value to enable more verbose messaging. Set to zero (default) to turn off verbose messaging.

Cube Manipulation TurboIntegrator Functions

These functions pertain to manipulating cubes.

AddCubeDependency

AddCubeDependency lets you predefine cube inter-dependencies to avoid lock contention problems during normal system use.

In normal operations, cube dependencies are established when data which crosses cube boundaries (such as data that is derived by a rule that references an external cube) is retrieved. To create the dependency information, the server must lock the cubes while the dependency is established, potentially maintaining the lock during a long view calculation. Since this is a 'write' lock, other users are prevented from accessing the cubes. The AddCubeDependency function allows the dependency to be established when the server starts up, preventing later lock contention as no new dependency need be established.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AddCubeDependency(BaseCube, DependentCube);
```

Argument	Description
BaseCube	The name of the cube upon which the DependentCube is dependent.
DependentCube	The name of a cube that depends on another cube (BaseCube) for data. Most commonly, this would be a cube that uses rules to pull data from an external cube.

Example

Consider a cube named 'SalesCube' that includes the rule `['net']=!Units * DB('PriceCube', ... );`

In this example, 'SalesCube' is the dependent cube, as it is dependent on values in the base cube named 'PriceCube' to calculate the value of 'net'. To establish this dependency, you should run the following function in a TurboIntegrator process: `AddCubeDependency( 'PriceCube', 'SalesCube' );`

To establish dependency at server load time, you can create a process that runs the `AddCubeDependency` function, schedule the process as a chore, and then define that chore as one of the **`StartupChores`** in `Tm1s.cfg`.

CellGetN

CellGetN retrieves a value from a numeric cube cell.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellGetN(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube from which you want to retrieve a value.
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables. If any of the dimensions in your cube use hierarchies, you can also use the `'HierarchyName':'ElementName'` convention to specify an element within a specific hierarchy. In this case, the sequence of arguments must still adhere to the order of dimensions in the cube. For example, if you want to reference multiple elements from hierarchies in the second dimension of your cube, all such arguments must appear after the argument for the first dimension in the cube and before the argument for the third dimension in the cube.

Example of CELLGETN and variables

When this function is used in a conditional statement (`IF`), the statement is the portion containing the condition, not the entire conditional block. After a minor error, execution continues with the next statement. TurboIntegrator processing has no knowledge that it was in a conditional once the minor error is processed, so the next statement is the next line, not the line after the `ENDIF;`.

To avoid this situation, use variables for any operation that could encounter a minor error and then use the variables in the conditional statement.

```
V1 = CELLGETN('PNLCube', 'fred','argentina','Sales','Jan');
IF(V1 = 454);
    ASCIIOUTPUT('bug.txt', 'if logic not working properly');
ENDIF;
```

Example of CellGetN without hierarchies

This example illustrates a function that uses simple element names as arguments and does not reference any hierarchies. It retrieves the numeric value at the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the Sales cube.

```
CellGetN('Sales', 'Actual', 'Argentina', 'S Series1.8L Sedan', 'Sales', 'Jan');
```

Example of CellGetN referencing multiple hierarchies

This example illustrates a function that references multiple hierarchies in the Model dimension. It retrieves the numeric value at the intersection of the Actual, Argentina, S Series (from the vehicles hierarchy in the model dimension), 2.8 Litre (from the enginesize hierarchy in the model dimension), and Jan elements in the Sales cube.

```
CellGetN('Sales', 'Actual', 'Argentina',
('vehicles':'S Series', 'enginesize':'2.8 Litre'), 'Sales', 'Jan');
```

Note: This example artificially breaks the line of code for easier reading.

CellGetS

CellGetS retrieves a value from a string cube cell.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellGetS(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube from which you want to retrieve a value.
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

See the note at "CellGetN" on page 485 concerning IF logic with this function.

Example

```
CellGetS('Personnel', 'Rep', 'Europe', 'Product');
```

This example retrieves the string value at the intersection of the Rep, Europe, and Product elements in the Personnel cube.

CellIncrementN

CellIncrementN increments an existing numeric cell value by a specified value.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellIncrementN(x, Cube, e1, e2 [,...en]);
```

Argument	Description
x	A numeric value that you want to add to an existing cell value.

Argument	Description
Cube	The name of the cube to which you want to send the value.
e1,...e*n*	Dimension element names that define the intersection of the cube to receive the value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellIncrementN(1000, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example increments the value at the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the y2ksales cube by 1000.

CellIsUpdateable

CellIsUpdateable determines whether a cube cell can be written to. The function returns 1 if the cell can be written to, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellIsUpdateable(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube to which you want to write a value.
e1,...e*n*	Dimension element names that define the cell to which you want to write a value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellIsUpdateable ('y2ksales', 'Actual', 'Argentina','S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example determines if the cell defined by the elements Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan in the y2ksales cube can be written to. If the cell can receive a value, the function returns 1, otherwise it returns 0.

CellPutN

CellPutN sends a numeric value to a cube cell.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellPutN(x, Cube, e1, e2 [,..., en]);
```

Argument	Description
x	A numeric value.
Cube	The name of the cube to which you want to send the value.
e1, e2, ..., en	Dimension element names that define the intersection of the cube to receive the value. Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Note: If you supply invalid arguments to the `CellPutN()` function in a TurboIntegrator process when the cube does not exist, an error is sent to the `tm1server.log`.

Example

```
CellPutN(12345, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example sends the value 12345 to the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the y2ksales cube.

CellPutProportionalSpread

CellPutProportionalSpread distributes a specified value to the leaves of a consolidation proportional to existing cell values. CellPutProportionalSpread replaces existing cell values; it cannot be used to add to or subtract from existing cell values.

The function is analogous to the Proportional Spread data spreading method, which is described in detail in the *TM1 Perspectives and TM1 Architect* documentation. If you must add to or subtract from existing cell values, use the Proportional Spread method, which can be executed through the user interface or through data spreading syntax.

Note: When using CellPutProportionalSpread to distribute a value to the leaves of a consolidation, only those leaves already containing non-zero values are changed. This is because zero values cannot be incremented or decremented proportionally; any proportion of zero is still zero.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellPutProportionalSpread( value, cube, e1, e2, e3...,en );
```

Argument	Description
value	The value you want to distribute.
cube	The name of the cube into which you want to distribute the value.
e1...e*n*	The names of the elements that identify the consolidation whose leaves will accept the distributed value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellPutProportionalSpread(7000,'SalesCube', 'Actual','North America',
'S Series 1.8L Sedan', 'Sales', 'Jan')
```

This example distributes the value 7000 to the children of the consolidation in the SalesCube identified by the elements Actual, North America, S Series 1.8L Sedan, Sales, and Jan.

CellPutS

CellPutS sends a string value to a cube cell.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellPutS(String, Cube, e1, e2 [,...en]);
```

Argument	Description
String	A string.
Cube	The name of the cube to which you want to send the string.
e1,...e*n*	Dimension element names that define the intersection of the cube to receive the string. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellPutS('jones', 'Personnel', 'Rep', 'Europe', 'Product');
```

This example sends the string 'jones' to the intersection of the Rep, Europe, and Product elements in the personnel cube.

CubeClearData

CubeClearData clears all of the data in a cube. This function is much faster than doing an operation such as creating a view to cover the entire cube, and then doing a ViewZeroOut() to zero out the entire cube.

When you use CubeClearData to clear data from a cube, any cells in the cube that are fed with feeders are also cleared. You must resave the rule that establishes the feeders or use the CubeProcessFeeders function to restore the fed cells.

This function is valid in TM1 TurboIntegrator processes only.

Note: This call just deletes the cube data, it does not delete and re-create the cube itself. This has implications when sandboxes are used. If a cube is deleted and then re-created any sandboxes a user may have will be discarded, since the cube against which those sandboxes were created was deleted (even though a cube may have been re-created with the same name). If however the CubeClearData() call is used, the sandbox data will still be considered valid, since the cube against which the sandbox was created continues to exist.

Syntax

```
CubeClearData( name-of-cube-as-string );
```

Argument

The name of the cube to clear, as a string.

Example

```
CubeClearData( 'expense' );
```

CubeCreate

CubeCreate creates a cube from specified dimensions. The order of dimensions specified in the function will be the order of dimensions in the cube definition. After execution, CubeCreate automatically saves the resulting .cub file to disk.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeCreate(Cube, d1, d2 [,...dn]);
```

Argument	**Description**
Cube	The name you want to assign to the cube.
d1,...d*n*	The names of dimensions that comprise the cube. You must specify at least two, but no more than 256, dimensions.

Example

```
CubeCreate('y2ksales', 'Actvsbud', 'Region', 'Model','Account1', 'Month');
```

This example creates a cube named y2ksales using the dimensions Actvsbud, Region, Model, Account1, and Month.

CubeDestroy

CubeDestroy deletes a specified TM1 cube.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDestroy(Cube);
```

Argument	Description
Cube	The name of the cube you want to delete.

Example

```
CubeDestroy('y2ksales');
```

This example deletes the cube named y2ksales.

CubeDimensionCountGet

CubeDimensionCountGet returns the number of dimensions in a cube.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDimensionCountGet(CubeName);
```

Argument	Description
CubeName	The name of the cube for which you want to determine the number of dimensions.

Example

```
CubeDimensionCountGet('Sales');
```

In this example, the function returns the number of dimensions in the Sales cube.

CubeExists

CubeExists determines whether a specific cube exists on the server from which a TurboIntegrator process is executed. The function returns 1 if the cube exists on the server, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeExists(CubeName);
```

Argument	Description
CubeName	The name of the cube whose existence you want to confirm.

Example

```
CubeExists('Inventory');
```

This example determines if the Inventory cube exists on the server.

CubeGetLogChanges

CubeGetLogChanges returns the Boolean value of the Logging property for a specified cube.

The Logging property is set in the TM1 Security Assignments dialog box and stored in the }CubeProperties control cube. If Logging is turned on for a cube, the function returns 1. If logging is turned off the function returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeGetLogChanges(CubeName);
```

Argument	Description
CubeName	The cube for which you want to return the value of the Logging property.

Example

```
CubeGetLogChanges('2002sales');
```

If Logging is turned on for the 2002sales cube, the function returns 1.

CubeSaveData

CubeSaveData() serializes a cube.

This function is valid in TM1 TurboIntegrator processes only.

To improve performance, transaction logging may be disabled while loading data. To safeguard newly loaded data in the unlikely event of a server crash, the changes can be serialized to disk. SaveDataAll has been used to serialize data to disk and to truncate the transaction log. When processing a SaveDataAll command, the server acquires a READ lock on every cube and an IX lock on every changed cube. This can cause significant contention with user activity if SaveDataAll is run during periods of user activity.

Typically not all the cubes affected by SaveDataAll need to be serialized since not all cubes are typically loaded with new data. CubeSaveData is used to serialize an individual cube to disk. CubeSaveData serializes the cube's data that has been committed to memory including the modifications that have been performed against it in the current TurboIntegrator process but not yet committed.

Syntax

```
CubeSaveData(Cube);
```

Argument	Description
Cube	The name of the cube you want to serialize.

Example

```
CubeSaveData ('SalesCube');
```

Consider the following TurboIntegrator process code:

```
CellPutN(500, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');

CubeSaveData('y2ksales');
```

```
CellPutN(1000, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');
```

When the CubeSaveData command is processed, the value of 500 for the January Sales cell will be included in the cube's serialization to disk, even though it has not yet been committed. The update of the January Sales cell to 1000 will not be part of the serialization.

Transaction Log

A new transaction entry appears in the Transaction log when CubeSaveData has been run. When processing a transaction log file during recovery, all updates to a cube that have been applied so far will be discarded when a CubeSaveData directive against the cube is encountered as all of the updates have already been serialized to the cube.

Server Crash Recovery

The SaveDataAll command takes advantage of the fact that all cubes are locked during its processing and truncates the transaction log knowing that all updates performed before serialization have been safely stored to disk. This is not the case for CubeSaveData so you must modify the way data recovery is performed when a cube has been serialized.

The transaction log file could contain records that represent changes that are older than the most recent data in the cube and should not be applied when data is being recovered.

CubeSetConnParams

CubeSetConnParams is used to encrypt the password for a virtual cube in the }CubeProperties cube.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeSetConnParams(cubeName, providerName, dataSourceLocation,dataSourceName,
dataSourceCatalog, userID, password, sapClientID, sapClientLang, providerString);
```

Argument	Description
cubeName	The name of the cube for which you want to set the password.
providerName	
dataSourceLocation	Name your administrator assigns to a set of catalogs at a particular location. In Microsoft Analysis Services, this is the name of a registered server.
dataSourceName	
dataSource catalog	The name assigned by your administrator to a particular collection of databases (Cubes, Dimensions and other objects). For MAS, this is the name of the database.
UserID	A valid username for the database.
Password	Password to use for this data source.
sapClientID	SAP client ID

Argument	Description
sapClientLang	SAP language setting.
providerString	

Example

```
CubeSetConnParams(sc, TM1OLAP, tm1server, , sdata, admin, apple, , , );
```

CubeSetLogChanges

CubeSetLogChanges sets the Logging property for a cube.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeSetLogChanges(Cube, LogChanges);
```

Argument	Description
Cube	The name of the cube for which you want to set the LOGGING property.
LogChanges	The Boolean value you want to assign to the property. 1= LOGGING on, 0 = LOGGING off.

CubeTimeLastUpdated

CubeTimeLastUpdated returns a serial value that indicates the date and time at which a specified cube was last updated.

The serial value that is returned by this function uses a starting time of Jan 1 1900 12:00:00 A.M., which is equivalent to the value 1.0. Dates are represented by integers, while times are represented as decimal numbers between .0 and .999999. This is consistent with the way date and time serial values are stored and reported in Microsoft Excel.

Note: By default, TM1 date and time serial values use a starting time of Jan 1 1960 12:00:00 A.M. To resolve the inconsistency between Excel and TM1 date and time serial values, you can set **`UseExcelSerialDate=T`** in your Tm1s.cfg file to instruct the TM1 server to use date and time serial values that conform to Excel standards.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeTimeLastUpdated(cube);
```

Argument	Description
cube	The name of the cube.

Example

```
CubeTimeLastUpdated('Sales');
```

This example returns a value corresponding to the time when the Sales cube was last updated.

CubeUnload

CubeUnload unloads a specified cube, along with all associated cube views, from memory.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeUnload(CubeName);
```

Argument	Description
CubeName	The cube you want to unload from memory.

Example

```
CubeUnload('ManufacturingBudget');
```

This example unloads the ManufacturingBudget cube, and any associated views, from server memory.

Data Reservation TurboIntegrator Functions

Use the following TurboIntegrator functions to programmatically obtain, release and manage Data Reservations.

For more details about using the Data Reservation feature, see "Using Data Reservations" in the IBM Cognos *TM1 for Developers* documentation.

CubeDataReservationAcquire

CubeDataReservationAcquire acquires a Data Reservation for the specified cube, user and tuple.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDataReservationAcquire(Cube, User, bForce, Address, [AddressDelimiter])
```

Argument	Description
Cube	Name of the cube.
User	Name of the owner for the new reservation. The user name supplied will be validated to make sure it is an existing user.
bForce	Boolean value that determines the behavior if the requested reservation conflicts with an existing reservation. If set to 0 (false), then the request is rejected if it conflicts with an existing reservation. If set to 1 (true) and the user running the TurboIntegrator process has the DataReservationOverride capability, then the conflicting reservations are released, and the requested one is granted.

Argument	Description
Address	Tokenized string sequence of element names that define the tuple. The order must match the original dimension order of the cube. All the cells in the cube contained by the tuple make up the region being reserved. You can choose one element from each dimension or use an empty string between the delimiters to select an entire dimension. Depending on where the element is located in the hierarchy, the request reserves a single cell, a slice, or the entire cube.
AddressDelimiter	Optional character string that is used to separate element names in the Address parameter. Default value is '|'.

Return Value

Boolean - returns true if the acquisition succeeded.

Example

```
CubeDataReservationAcquire('DRTestCube','User1',0,'ElemX|ElemY|ElemZ');
```

The following example sets the bForce parameter to 1 to force the DR request if a conflict exists and uses a different delimiter character for the AddressDelimiter parameter.

```
CubeDataReservationAcquire('DRTestCube','User2',1,'ElemX*ElemY*ElemZ','*');
```

CubeDataReservationRelease

CubeDataReservationRelease releases the specified Data Reservation.

If the user specified is not the same as the owner of the reservation, then the release will only succeed if the user specified has the DataReservationOverride capability enabled.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDataReservationRelease(Cube, User, Address,[AddressDelimiter])
```

Argument	Description
Cube	Name of the cube.
User	Name of the owner of the reservation. The user name supplied will be validated to make sure it is an existing user.
Address	Tokenized string sequence of element names that define the tuple. The order must match the original dimension order of the cube.
AddressDelimiter	Optional character string that is used to separate element names in the Address parameter. Default value is '|'.

Return Value

Boolean - returns true if the release succeeded.

Example

```
CubeDataReservationRelease('DRTestCube','User1','ElemX|ElemY|ElemZ');
```

The following example uses a different character for the AddressDelimiter parameter.

```
CubeDataReservationRelease('DRTestCube','User2','ElemX*ElemY*ElemZ','*');
```

CubeDataReservationReleaseAll

CubeDataReservationReleaseAll releases multiple existing Data Reservations.

All reservations fully contained by the specified address that match the user filter will be released. A blank user filter means all users.

If the user filter specified is not the same as the user running the TurboIntegrator proces, then the DataReservationOverride capability must be enabled.

Using a blank user filter and all wildcards in the address field releases all reservations.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDataReservationReleaseAll(Cube, UserFilter, Address, [AddressDelimiter])
```

Argument	Description
Cube	Name of the cube.
UserFilter	User name filter to match against existing reservations.
Address	Tokenized string sequence of element names that define the tuple. The order must match the original dimension order of the cube.
AddressDelimiter	Optional character string that is used to separate element names in the Address parameter. Default value is '|'.

Return Value

Boolean - returns true if no errors.

Example

```
CubeDataReservationReleaseAll('DRTestCube','User1','ElemX|ElemY|ElemZ');
```

The following example releases all reservations in the specified cube for all users.

```
CubeDataReservationReleaseAll('DRTestCube','','||');
```

CubeDataReservationGet

CubeDataReservationGet finds existing reservations on a specific cube for all or one user.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDataReservationGet(Index, Cube, User, [AddressDelimiter]) returns Address;
```

Argument	Description
Index	A one-based loop index to use for iterating through reservations on the specified cube.
Cube	Name of the cube to search.
User	Reservation owner name to use as a filter. If left blank, the function returns reservations for any owner. If a name is provided, the function filters the results for just the specified owner.
AddressDelimiter	Optional character string that is used to separate element names in the returned Address parameter. Default value is '|'.

Return Value

Address - Reservation creation time, name of the reservation owner and Element address of the reservation. Creation time comes first, followed by delimiter, followed by UserID, followed by delimiter, followed by Elements IDs separated by the delimiter in order of dimensions in the cube (original order).

An empty string is returned if there is no entry for the specified index.

The format of the return value is:

`[creation time][delimiter][owner name][delimiter][element1][delimiter][element2][delimiter]…[elementN]`

For example:

"20100622211601|Fred Bloggs|Element1|Element2|Element3"

Note: The reservations can change while iterating the list of reservations so the use of index is not guaranteed to give a complete list of reservations. Reservations can be added or removed at any position in the list, so reservations can be skipped or repeated when looping through index values.

If the owner filter is specified, then the index applies only to the members of the filtered list. If the list of reservations has owners as follows: User1, User1, User2 and the request specifies an owner of User2 then an index of 1 will retrieve the third member of the list.

Example

```
CubeDataReservationGet(1,'DRTestCube','User1','*');
```

```
CubeDataReservationGet(1,'DRTestCube','');
```

The following sample would find all the reservations owned by user Fred Bloggs in the Expense Input cube and do "something useful" with them:

```
vIndex = 1;
vCube = 'Expense Input';
vUserFilter = 'Fred Bloggs';
vDelim = '|';
vAddress = CubeDataReservationGet( vIndex, vCube, vUserFilter,vDelim);
WHILE (vAddress @<> '');
    vSep1 = SCAN( vDelim, vAddress);
```

```
        vDRUser = SUBST( vAddress, 1, vSep1 - 1);
        vDRAddress = SUBST( vAddress, vSep1 + 1, LONG(vDRAddress) - vSep1);

#     do something meaningful with the
user and reservation address here
        vIndex = vIndex + 1;
        vAddress = CubeDataReservationGet( vIndex, vCube, vUserFilter,vDelim);
END;
```

CubeDataReservationGetConflicts

CubeDataReservationGetConflicts finds existing reservations on a specific cube that would conflict with the specified user, address and tuple.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDataReservationGetConflicts(Index, Cube, User, Address, [AddressDelimiter])returns
ConflictAddress;
```

Argument	Description
Index	A one-based loop index to use for iterating through conflicts that satisfy this query.
Cube	Name of the cube to search
User	The query will search for reservations that will conflict with this user.
Address	Tokenized string sequence of element names that define the tuple. The order must match the original dimension order of the cube.
AddressDelimiter	Optional character string that is used to separate element names in the Address parameter. Default value '|'.

Return Value

ConflictAddress - Reservation creation time, name of the reservation owner and Element address of the reservation. The creation time comes first, followed by delimiter, followed by UserID, followed by delimiter, followed by Elements IDs separated by the delimiter in order of dimensions in the cube (original order).

An empty string is returned if there is no entry for the specified index.

The format of the return value is:

```
                [creation time][delimiter][owner name][delimiter][element1][delimiter]
[element2][delimiter]…[elementN]
```

For example:

"20100622211601|Fred Bloggs|Element1|Element2|Element3"

Note: The reservations can change while iterating the list of conflict reservations so the use of index is not guaranteed to give a complete list of reservations. Reservations can be added or removed at any position in the list, so reservations can be skipped or repeated when looping through index values.

CubeDRAcquire

CubeDRAcquire acquires a Data Reservation for the specified cube, user and tuple. While the CubeDataReservationAcquire applies to dimensions with a single hierarchy, this function applies to dimensions with one or more hierarchies.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDRAcquire(Cube, User, bForce, Element-list)
```

Argument	Description
Cube	Name of the cube.
User	Name of the owner for the new reservation. The user name supplied will be validated to make sure it is an existing user.
bForce	Boolean value that determines the behavior if the requested reservation conflicts with an existing reservation. If set to 0 (false), then the request is rejected if it conflicts with an existing reservation. If set to 1 (true) and the user running the TurboIntegrator process has the DataReservationOverride capability, then the conflicting reservations are released, and the requested one is granted.
Element-list	Coordinates are expressed as a variable-length list of individual element name arguments. Each of the element name parameters is expected to be hierarchy-qualified. If the dimension has one hierarchy only, then use the name of the dimension. Otherwise, use the name of the intended hierarchy. Elements are specified in a '<hierarchy>':'<element>' format, with each instance separated by a comma.

Return Value

Boolean - returns true if the acquisition succeeded.

Example

```
CubeDRAcquire('DRTestCube','User1',0,'Hier1':'ElemX','Hier2':'ElemY','Hier3':'ElemZ');
```

The following example sets the bForce parameter to 1 to force the DR request if a conflict exists and uses a different delimiter character for the AddressDelimiter parameter.

```
CubeDRAcquire('DRTestCube','User2',1,'Hier1':'ElemX','Hier2':'ElemY','Hier3':'ElemZ');
```

CubeDRRelease

CubeDRRelease releases the specified Data Reservation. While the CubeDataReservationRelease applies to dimensions with a single hierarchy, this function applies to dimensions with one or more hierarchies.

If the user specified is not the same as the owner of the reservation, then the release will only succeed if the user specified has the DataReservationOverride capability enabled.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDRRelease(Cube, User, Element-list)
```

Argument	Description
Cube	Name of the cube.
User	Name of the owner of the reservation. The user name supplied will be validated to make sure it is an existing user.
Element-list	Coordinates are expressed as a variable-length list of individual element name arguments. Each of the element name parameters is expected to be hierarchy-qualified. If the dimension has one hierarchy only, then use the name of the dimension. Otherwise, use the name of the intended hierarchy. Elements are specified in a '<hierarchy>':'<element>' format, with each instance separated by a comma.

Return Value

Boolean - returns true if the release succeeded.

Example

```
CubeDRRelease('DRTestCube','User1','Hier1':'ElemX','Hier2':'ElemY','Hier3':'ElemZ');
```

The following example uses a different character for the AddressDelimiter parameter.

```
CubeDRRelease('DRTestCube','User2','Hier1':'ElemX','Hier2':'ElemY','Hier3':'ElemZ');
```

CubeDRReleaseAll

CubeDRReleaseAll releases multiple existing Data Reservations. While the CubeDataReservationReleaseAll applies to dimensions with a single hierarchy, this function applies to dimensions with one or more hierarchies.

All reservations fully contained by the specified address that match the user filter will be released. A blank user filter means all users.

If the user filter specified is not the same as the user running the TurboIntegrator proces, then the DataReservationOverride capability must be enabled.

Using a blank user filter and all wildcards in the address field releases all reservations.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDRReleaseAll(Cube, UserFilter, Element-list)
```

Argument	Description
Cube	Name of the cube.
UserFilter	User name filter to match against existing reservations.

Argument	Description
Element-list	Coordinates are expressed as a variable-length list of individual element name arguments. Each of the element name parameters is expected to be hierarchy-qualified. If the dimension has one hierarchy only, then use the name of the dimension. Otherwise, use the name of the intended hierarchy. Elements are specified in a '<hierarchy>':'<element>' format, with each instance separated by a comma.

Return Value

Boolean - returns true if no errors.

Example

```
CubeDRReleaseAll('DRTestCube','User1','Hier1':'ElemX','Hier2':'ElemY','Hier3':'ElemZ');
```

The following example releases all reservations in the specified cube for all users.

```
CubeDRReleaseAll('DRTestCube','','');
```

CubeDRGet

CubeDRGet finds existing reservations on a specific cube for all or one user. While the CubeDataReservationGet applies to dimensions with a single hierarchy, this function applies to dimensions with one or more hierarchies.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDRGet(Index, Cube, User, Element-list) returns Address;
```

Argument	Description
Index	A one-based loop index to use for iterating through reservations on the specified cube.
Cube	Name of the cube to search.
User	Reservation owner name to use as a filter. If left blank, the function returns reservations for any owner. If a name is provided, the function filters the results for just the specified owner.
Element-list	Coordinates are expressed as a variable-length list of individual element name arguments. Each of the element name parameters is expected to be hierarchy-qualified. If the dimension has one hierarchy only, then use the name of the dimension. Otherwise, use the name of the intended hierarchy. Elements are specified in a '<hierarchy>':'<element>' format, with each instance separated by a comma.

Return Value

Address - Reservation creation time, name of the reservation owner and Element address of the reservation. Creation time comes first, followed by delimiter, followed by UserID, followed by delimiter, followed by Elements IDs separated by the delimiter in order of dimensions in the cube (original order).

An empty string is returned if there is no entry for the specified index.

The format of the return value is:

```
[creation time][delimiter][owner name][delimiter][element1][delimiter]
[element2][delimiter]…[elementN]
```

For example:

"20100622211601|Fred Bloggs|Element1|Element2|Element3"

Note: The reservations can change while iterating the list of reservations so the use of index is not guaranteed to give a complete list of reservations. Reservations can be added or removed at any position in the list, so reservations can be skipped or repeated when looping through index values.

If the owner filter is specified, then the index applies only to the members of the filtered list. If the list of reservations has owners as follows: User1, User1, User2 and the request specifies an owner of User2 then an index of 1 will retrieve the third member of the list.

Example

```
CubeDRGet(1,'DRTestCube','User1','*');
```

```
CubeDRGet(1,'DRTestCube','');
```

The following sample would find all the reservations owned by user Fred Bloggs in the Expense Input cube and do "something useful" with them:

```
vIndex = 1;
vCube = 'Expense Input';
vUserFilter = 'Fred Bloggs';
vHier = 'Currency';
vElem = 'Local Currency';
vAddress = CubeDRGet( vIndex, vCube, vUserFilter, vHier:vElem);
WHILE (vAddress @<> '');
    vSep1 = SCAN( vHier:vElem, vAddress);
        vDRUser = SUBST( vAddress, 1, vSep1 - 1);
        vDRAddress = SUBST( vAddress, vSep1 + 1, LONG(vDRAddress) - vSep1);

#     do something meaningful with the
user and reservation address here
        vIndex = vIndex + 1;
        vAddress = CubeDRGet( vIndex, vCube, vUserFilter, vHier:vElem);
END;
```

CubeDRGetConflicts

CubeDRGetConflicts finds existing reservations on a specific cube that would conflict with the specified user. While the CubeDataReservationGetConflicts applies to dimensions with a single hierarchy, this function applies to dimensions with one or more hierarchies.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeDRGetConflicts(Index, Cube, User, Element-list) returns ConflictAddress;
```

Argument	Description
Index	A one-based loop index to use for iterating through conflicts that satisfy this query.
Cube	Name of the cube to search
User	The query will search for reservations that will conflict with this user.
Element-list	Coordinates are expressed as a variable-length list of individual element name arguments. Each of the element name parameters is expected to be hierarchy-qualified. If the dimension has one hierarchy only, then use the name of the dimension. Otherwise, use the name of the intended hierarchy. Elements are specified in a '<hierarchy>':'<element>' format, with each instance separated by a comma.

Return Value

ConflictAddress - Reservation creation time, name of the reservation owner and Element address of the reservation. The creation time comes first, followed by delimiter, followed by UserID, followed by delimiter, followed by Elements IDs separated by the delimiter in order of dimensions in the cube (original order).

An empty string is returned if there is no entry for the specified index.

The format of the return value is:

```
[creation time][delimiter][owner name][delimiter][element1][delimiter]
[element2][delimiter]…[elementN]
```

For example:

"20100622211601|Fred Bloggs|Element1|Element2|Element3"

Note: The reservations can change while iterating the list of conflict reservations so the use of index is not guaranteed to give a complete list of reservations. Reservations can be added or removed at any position in the list, so reservations can be skipped or repeated when looping through index values.

Date and Time TurboIntegrator Functions

These functions format and parse dates and times in a wide variety of formats and locales.

FormatDate

FormatDate formats a date value according to a formatter defined with the NewDateFormatter function.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
FormatDate(Date, <Pattern>, <Index>)
```

Argument	Description
Date	A date value. The type of value, serial or UNIX, should correspond to the formatter used.

Argument	Description
Pattern	Pattern used for formatting dates. Refer to http://userguide.icu-project.org/formatparse/datetime for a complete list of format syntax. If an empty string is passed, then the format is determined by the locale based on the FormatterStyle and FormatterType parameters that were used with the NewDateFormatter function.
Index	Index returned by a call to the NewDateFormatter function. The default value is 0. If no date formatter exists at the index, then a default formatter is used as though it had been created with the following call: `NewDateFormatter('', 'Etc/UTC', 'serial', 'medium', 'date')`

Example

```
sDate = FormatDate(18000);
```

NewDateFormatter

NewDateFormatter defines a date formatter. It returns an index for use in the ParseDate and FormatDate functions. The indices start at 0 and go up by one for each call to NewDateFormat. Date formatters are valid during execution of the process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
NewDateFormatter(Locale, <TimeZone>, <UseUNIXTime>, <FormatterStyle>, <FormatterType>,
<TimeType>)
```

Argument	Description
Locale	Locale used for parsing or formatting dates. If an empty string is passed, then the operating system locale is used. Locales are specified in the format language[_territory][.variant]. For example, `cs_CK` is the Czech language and Czech Republic.
TimeZone	Timezone used for parsing or formatting dates. Refer to http://en.wikipedia.org/wiki/List_of_tz_database_time_zones for a complete list of time zones. If not specified, the time zone used is `UTC ('Etc/UTC')`.
UseUNIXTime	If `'unix'` is specified, then times are treated as milliseconds since January 1, 1970. Otherwise, they are treated in TM1 serial format. Note that only dates later than January 1, 1970 can be processed even if TM1 serial format is used.

Argument	Description
FormatterStyle	Controls the date format used when an empty pattern is specified to the FormatDate or ParseDate functions. Valid values are `'full'`, `'long'`, `'medium'` or `'short'`. The default is `'medium'`.
FormatterType	Controls the type of format used when an empty pattern is specified to the FormatDate or ParseDate functions. Valid values are `'time'`, `'date'` or `'datetime'`. The default is `'date'`.

Example

```
dfUNIX = NewDateFormatter('', 'Etc/UTC', 'unix');
```

```
dfStyleFullDateTime = NewDateFormatter('en_us', 'America/Toronto', 'serial', 'full',
'datetime');
```

ParseDate

ParseDate parses a date string according to a formatter defined with the NewDateFormatter function.

A date value that is either serial or UNIX, depending on the formatter specified, is returned. If the date cannot be parsed then an undefined value is returned. This can be tested with the ISUND function.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ParseDate (DateString, <Pattern>, <Index>)
```

Argument	Description
DateString	A date string.
Pattern	Pattern used for parsing dates. Refer to http://userguide.icu-project.org/formatparse/datetime for a complete list of format syntax. If an empty string is passed, then the format is determined by the locale based on the FormatterStyle and FormatterType parameters that were used with the NewDateFormatter function.
Index	Index returned by a call to the NewDateFormatter function. The default value is 0. If no date formatter exists at the index, then a default formatter is used as though it had been created with the following call: `NewDateFormatter('', 'Etc/UTC', 'serial', 'medium', 'date')`

Example

```
nDate = ParseDate('2011/11/24', 'yyyy/MM/dd');
```

Dimension Manipulation TurboIntegrator Functions

These functions facilitate the manipulation of dimensions.

DimensionCreate

DimensionCreate creates a new dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionCreate(DimName);
```

Argument	Description
DimName	The name you want to assign to the dimension.

Example

```
DimensionCreate('Product');
```

This example creates the Product dimension.

DimensionDeleteAllElements

DimensionDeleteAllElements deletes all the elements in a dimension. This function is useful for recreating dimension hierarchies.

Note: Deleting an element deletes all cube data identified by that element. However, if you use DimensionDeleteAllElements to delete elements, then recreate those elements with the same names in the Metadata tab, any data points in a cube identified by the elements will be retained after rebuilding the dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionDeleteAllElements(DimName);
```

Argument	Description
DimName	The name of the dimension from which you want to delete all elements.

Example

```
DimensionDeleteAllElements('Model');
```

This example deletes all elements in the Model dimension.

DimensionDeleteElements

DimensionDeleteElements deletes all elements from a dimension using the subset of elements. All elements in the referenced subset are deleted, including C level elements.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionDeleteElements (DimensionName, Subset )
```

Argument	Description
DimensionName	The name of the dimension from which you want to delete the subset of elements.
Subset	The list of elements to delete from the indicated dimension. The subset is usually temporary.

DimensionDestroy

DimensionDestroy deletes a dimension from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionDestroy(DimName);
```

Argument	Description
DimName	The name of the dimension you want to delete.

Example

```
DimensionDestroy('Product');
```

This example deletes the Product dimension from the TM1 database.

DimensionElementComponentAdd

DimensionElementComponentAdd adds a component (child) to a consolidated element. You can't use this function in the Epilog procedure of a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementComponentAdd(DimName, ConsolidatedElName,ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
ConsolidatedElName	The element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
DimensionElementComponentAdd('Measures', 'Net Sales', 'Expenses', -1);
```

This example adds the child Expenses to the Net Sales consolidation in the Measures dimension. The child has a weight of -1 in the consolidation.

DimensionElementComponentAddDirect

DimensionElementComponentAddDirect adds a component (child) to a consolidated element by directly editing a dimension.

This function is valid in TM1 TurboIntegrator processes only.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementComponentAdd) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementComponentAddDirect(DimName, ConsolidatedElName,ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
ConsolidatedElName	The consolidated element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
DimensionElementComponentAddDirect('Measures', 'Net Sales', 'Expenses', -1);
```

This example adds the child Expenses to the Net Sales consolidation in the Measures dimension. The child has a weight of -1 in the consolidation.

DimensionElementComponentDelete

DimensionElementComponentDelete deletes a component (child) from a consolidated element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementComponentDelete(DimName, ConsolidatedElName,ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
DimensionElementComponentDelete('Region', 'Benelux','Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Region dimension.

DimensionElementComponentDeleteDirect
DimensionElementComponentDeleteDirect deletes a component (child) from a consolidated element by directly editing the dimension.

This function is valid in TM1 TurboIntegrator processes only.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementComponentDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementComponentDeleteDirect(DimName, ConsolidatedElName,ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
DimensionElementComponentDeleteDirect('Region', 'Benelux','Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Region dimension.

DimensionElementDelete

DimensionElementDelete deletes an element from a dimension.

Note: Deleting an element deletes all cube data identified by that element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementDelete(DimName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
ElName	The element you want to delete.

Example

```
DimensionElementDelete('Region', 'Belgium');
```

This example deletes the element Belgium from the Region dimension.

DimensionElementDeleteDirect

DimensionElementDeleteDirect deletes an element from a dimension by directly editing the dimension.

This function is valid in TM1 TurboIntegrator processes only.

Note: Deleting an element deletes all cube data identified by that element.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementDeleteDirect(DimName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
ElName	The element you want to delete.

Example

```
DimensionElementDeleteDirect('Region', 'Belgium');
```

This example deletes the element Belgium from the Region dimension.

DimensionElementExists

DimensionElementExists determines whether a specific element exists in a dimension on the server from which a TurboIntegrator process is executed. The function returns 1 if the element exists in the dimension on the server, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementExists(DimName, ElName);
```

Argument	Description
DimName	The dimension that contains the element that you want to find. The dimension must exist on the server where the TurboIntegrator process is executed.
ElName	The element that you want to find.

Example

This example determines whether the element Belgium exists in the Region dimension on the server.

```
DimensionElementExists('Region', 'Belgium');
```

DimensionElementInsert

DimensionElementInsert adds an element to a dimension. You can use this function to add numeric, string, or consolidated elements. You can't use this function in the Data or Epilog procedures of a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementInsert(DimName, InsertionPoint, ElName,ElType);
```

Argument	Description
DimName	The dimension to which you want to add an element.
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is an empty string (''), the new element is added to the end of the dimension.
ElName	The name that you want to assign to the new element.
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
DimensionElementInsert('Region','Belgium','Netherlands','S');
```

This example adds the string element Netherlands to the Region dimension. Netherlands is added immediately before Belgium in the dimension.

Example with an empty insertion point

```
DimensionElementInsert('Region','','Netherlands','S');
```

This example adds the string element Netherlands to the Region dimension. Netherlands is added to the end of the dimension.

DimensionElementInsertDirect
DimensionElementInsertDirect adds an element to a dimension by directly editing the dimension. You can use this function to add numeric, string, or consolidated elements.

This function is valid in TM1 TurboIntegrator processes only.

The default method of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementInsert) are used in the metadata tab of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element insertion needed to support data loading is performed using direct calls in the Data procedure. When the Metadata

procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementInsertDirect(DimName, InsertionPoint, ElName,ElType);
```

Argument	Description
DimName	The dimension to which you want to add a new element.
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is empty, the new element is added to the end of the dimension. Note that this function is optimized for the case where the InsertionPoint is passed as an empty string.
ElName	The name you want to assign to the new element.
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
DimensionElementInsertDirect('Region', 'Belgium', 'Netherlands','N');
```

This example adds the numeric element Netherlands to the Region dimension. Netherlands displays immediately before Belgium in the dimension definition.

DimensionElementPrincipalName

DimensionElementPrincipalName returns the principal name of an element or element alias.

TurboIntegrator must use principal element names when updating dimensions; element aliases cannot be used. This function is useful for determining principal element names while attempting to update a dimension when only element aliases are available to the TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionElementPrincipalName( DimName, ElName )
```

Argument	Description
DimName	The name of the dimension from which you want to retrieve a principal element name.

Argument	Description
ElName	An element name. ElName can be either an element alias or a principal element name.

Example

If ElName is not in the currently saved version of DimName, the function returns ElName.

If ElName is in DimName, whether as an element alias or a principal element name, it returns the principal name of the element.

DimensionExists

DimensionExists determines whether a specific dimension exists on the server from which a TurboIntegrator process is executed. The function returns 1 if the dimension exists on the server, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionExists(DimName);
```

Argument	Description
DimName	The name of the dimension whose existence you want to confirm.

Example

```
DimensionExists('Region');
```

This example determines if the Region dimension exists on the server.

DimensionHierarchyCreate

DimensionHierarchyCreate creates a new hierarchy in an existing dimension. The hierarchy cannot have the same name as the dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionHierarchyCreate(DimName, HierName);
```

Argument	Description
DimName	The name of the existing dimension that will contain the hierarchy.
HierName	The name that you want to assign to the hierarchy. You cannot use the name of the dimension.

Example

```
DimensionHierarchyCreate('Vehicles', 'Trucks');
```

This example creates the empty Trucks hierarchy in the Vehicles dimension.

DimensionSortOrder

DimensionSortOrder sets a sort type and sense for dimension elements and for components of consolidated elements within a dimension. The sort order defined by DimensionSortOrder determines how the subset displays in the Subset Editor.

DimensionSortOrder sets properties for a dimension; the dimension is not actually sorted until it is saved on the server.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionSortOrder(DimName, CompSortType, CompSortSense, ElSortType , ElSortSense);
```

Argument	Description
DimName	The name of the dimension for which you want to set a sort order.
CompSortType	Defines how components of consolidated elements appear in the dimension. There are two CompSortType values: ByInput - Retains the order in which components were originally inserted into consolidations. ByName - Sorts components of consolidations by name.
CompSortSense	Defines the sort sense for components of consolidations. This is a required argument, but it applies only when the CompSortType is ByName. There are two possible CompSortSense values: Ascending - Sorts components of consolidations in ascending alphabetical order. Descending - Sorts components of consolidations in descending alphabetical order.
ElSortType	Defines a sort order for dimension elements. There are four possible ElSortType values: ByInput - Retains the order in which elements were originally inserted into the dimension. ByName - Sorts dimension elements by name. ByLevel - Sorts dimension elements by level. ByHierarchy - Sorts dimension elements by hierarchy.

Argument	Description
ElSortSense	Defines the sort sense for dimension elements. This is a required argument, but it applies only when the ElSortType is ByName or ByLevel. There are two possible ElSortSense values: Ascending - Sorts dimension elements in ascending order, either alphabetically or by level. Descending - Sorts dimension elements in descending order, either alphabetically or by level.

Example

```
DimensionSortOrder ('Region', 'ByName', 'Descending','ByLevel', 'Ascending');
```

This example sets a sort order for the Region dimension. All dimension elements are sorted by level in ascending order, and any components of consolidations are sorted in descending alphabetical order.

DimensionTimeLastUpdated

DimensionTimeLastUpdated returns a serial value that indicates the date and time at which a specified dimension was last updated.

The serial value returned by this function uses a starting time of Jan 1 1900 12:00:00 A.M., which is equivalent to the value 1.0. Dates are represented by integers, while times are represented as decimal numbers between .0 and .999999. This is consistent with the way date/time serial values are stored and reported in Microsoft Excel.

Note: By default, TM1 date/time serial values use a starting time of Jan 1 1960 12:00:00 A.M. To resolve the inconsistency between Excel and TM1 date/time serial values, you can set `UseExcelSerialDate=T` in your Tm1s.cfg file to instruct the TM1 server to use date/time serial values that conform to Excel standards.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionTimeLastUpdated(dimension);
```

Argument	Description
dimension	The name of the dimension.

Example

```
DimensionTimeLastUpdated('Region');
```

This example returns information on when the Region dimension was last updated.

DimensionTopElementInsert

DimensionTopElementInsert creates a root element in a dimension. If the dimension already has a single root, then this element will not be created.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionTopElementInsert(DimName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
DimensionTopElementInsert('Region', 'Netherlands', 'World');
```

This example adds the root element World to the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

DimensionTopElementInsertDirect

DimensionTopElementInsertDirect creates a root element in a dimension by directly editing the dimension. If the dimension already has a single root, then this element will not be created.

This function is valid in TM1 TurboIntegrator processes only.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionTopElementInsert) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionTopElementInsertDirect(DimName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.

Argument	Description
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
DimensionTopElementInsertDirect('Region', 'Netherlands', 'World');
```

This example adds the root element World to the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

DimensionUpdateDirect

DimensionUpdateDirect performs a full rewrite of a dimension that has been subject to direct editing in a TurboIntegrator process, essentially compacting the memory footprint of the dimension.

A dimension that undergoes a series of direct-only edits (element deletions, in particular) will eventually use more memory than its fully-rewritten counterpart would. This function can optionally be used after directly editing a dimension with DimensionElementInsertDirect, DimensionElementDeleteDirect, DimensionElementComponentAddDirect, DimensionElementComponentDeleteDirect, and/or DimensionTopElementInsertDirect. Calling DimensionUpdateDirect incurs an initial full-copy memory cost, however it can be used to guarantee that the dimension is at its smallest possible memory footprint after processing is complete.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DimensionUpdateDirect(DimName);
```

Argument	Description
DimName	The name of the dimension you want to rewrite.

Example

```
DimensionUpdateDirect('Region');
```

This example rewrites the Region dimension.

Hierarchy Manipulation TurboIntegrator Functions

These functions facilitate hierarchy manipulation.

HierarchyContainsAllLeaves

HierarchyContainsAllLeaves returns true only if the specified hierarchy contains the full set of leaf elements that are present in the dimension. That is, it contains all the leaf elements that can be seen in the special Leaves hierarchy. If the specified hierarchy is missing one or more leaf elements, this function returns false.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyContainsAllLeaves(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension that contains the all leaves hierarchy.
HierName	The name of the hierarchy you are determining as an all leaves hierarchy.

Example

```
HierarchyContainsAllLeaves('Region', 'Leaves');
```

This example determines if the Leaves hierarchy, in the Region dimension, contains all leaf members.

HierarchyCreate

HierarchyCreate creates a new hierarchy in an existing dimension. The hierarchy cannot have the same name as the dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyCreate(DimName, HierName);
```

Argument	Description
DimName	The name of the existing dimension that will contain the hierarchy.
HierName	The name you want to assign to the hierarchy. You cannot use the name of the dimension.

Example

```
HierarchyCreate('Vehicles', 'Trucks');
```

This example creates the empty Trucks hierarchy in the Vehicles dimension.

HierarchyDeleteAllElements

HierarchyDeleteAllElements deletes all the elements in a hierarchy. This function is useful for recreating dimension hierarchies.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyDeleteAllElements(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension from which you want to delete all elements.

Argument	Description
HierName	The name of the hierarchy within the dimension.

Example

```
HierarchyDeleteAllElements('Equipment','Helmets');
```

This example deletes all elements in the Helmets hierarchy in the Equipment dimension.

HierarchyDeleteElements

HierarchyDeleteElements deletes leaf elements from a hierarchy using a subset of elements.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyDeleteElements (DimensionName, HierarchyName, Subset)
```

Argument	Description
DimensionName	The name of the dimension from which you want to delete the subset of elements.
HierarchyName	The name of the hierarchy from which you want to delete the subset of elements. If the indicated hierarchy is the Leaves hierarchy, then the subset should list those leaves that should be deleted, and they are removed completely from the dimension.
Subset	The list of elements to delete from the indicated hierarchy. The subset is usually temporary.

HierarchyDestroy

HierarchyDestroy deletes a hierarchy from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyDestroy(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension that contains the hierarchy.
HierName	The name of the hierarchy you want to delete.

Example

```
HierarchyDestroy('Product','Transmissions');
```

This example deletes the Transmissions hierarchy from the TM1 database.

HierarchyElementComponentAdd

HierarchyElementComponentAdd adds a component (child) to a consolidated element. You can't use this function in the Epilog procedure of a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementComponentAdd(DimName, HierName, ConsolidatedElName, ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
HierName	The hierarchy of the specified dimension.
ConsolidatedElName	The element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
HierarchyElementComponentAdd('Measures', 'Europe', 'Net Sales', 'Expenses',
-1);
```

This example adds the child Expenses to the Net Sales consolidation in the Europe hierarchy of the Measures dimension. The child has a weight of -1 in the consolidation.

HierarchyElementComponentAddDirect

HierarchyElementComponentAddDirect adds a component (child) to a consolidated element by directly editing a dimension.

This function is valid in TM1 TurboIntegrator processes only.

The default method of editing a dimension in Cognos TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like HierarchyElementComponentAdd) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
HierarchyElementComponentAddDirect(DimName, HierName, ConsolidatedElName, ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
HierName	The hierarchy of the specified dimension.
ConsolidatedElName	The consolidated element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
HierarchyElementComponentAddDirect('Measures', 'Europe', 'Net Sales',
'Expenses', -1);
```

This example adds the child Expenses to the Net Sales consolidation in the Europe hierarchy of the Measures dimension. The child has a weight of -1 in the consolidation.

HierarchyElementComponentDelete

HierarchyElementComponentDelete deletes a component (child) from a consolidated element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementComponentDelete(DimName, HierName, ConsolidatedElName, ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
HierName	The name of the hierarchy within the dimension.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
HierarchyElementComponentDelete('Region', 'Western', 'Benelux', 'Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Western hierarchy of the Region dimension.

HierarchyElementComponentDeleteDirect

HierarchyElementComponentDeleteDirect deletes a component (child) from a consolidated element by directly editing the dimension.

This function is valid in TM1 TurboIntegrator processes only.

The default method of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like HierarchyElementComponentDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
HierarchyElementComponentDeleteDirect(DimName, HierName, ConsolidatedElName, ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
HierName	The name of the hierarchy within the dimension.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
HierarchyElementComponentDeleteDirect('Region', 'Western', 'Benelux', 'Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Western hierarchy of the Region dimension.

HierarchyElementDelete

HierarchyElementDelete deletes an element from a hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementDelete(DimName, HierName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
HierName	The name of the hierarchy within the dimension.
ElName	The element you want to delete from the hierarchy.

Example

```
HierarchyElementDelete('Region', 'Western', 'Belgium');
```

This example deletes the element Belgium from the Western hierarchy in the Region dimension.

HierarchyElementDeleteDirect

HierarchyElementDeleteDirect deletes an element from a dimension by directly editing the dimension.

This function is valid in TM1 TurboIntegrator processes only.

Note: Deleting an element deletes all cube data identified by that element.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
HierarchyElementDeleteDirect(DimName, HierName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
HierName	The name of the hierarchy within the dimension.
ElName	The element you want to delete.

Example

```
HierarchyElementDeleteDirect('Region', 'Western', 'Belgium');
```

This example deletes the element Belgium from the Western hierarchy in the Region dimension.

HierarchyElementExists
HierarchyElementExists determines whether a specific elements exists in a hierarchy on the server from which a TurboIntegrator process is executed. The function returns 1 if the elements exists in the hierarchy on the server, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementExists(DimName, HierName, ElemName);
```

Argument	Description
DimName	The name of the dimension that contains the element whose existence you want to confirm.
HierName	The name of the hierarchy within the dimension.
ElName	The element you want to find in the hierarchy.

Example

```
HierarchyElementExists('Region', 'Western', 'Belgium');
```

This example determines whether element Belgium from the Western hierarchy in the Region dimension exists on the server.

HierarchyElementInsert
HierarchyElementInsert adds an element to a dimension. You can use this function to add numeric, string, or consolidated elements. You can't use this function in the Data or Epilog procedures of a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementInsert(DimName, HierName, InsertionPoint, ElName, ElType);
```

Argument	Description
DimName	The dimension to which you want to add a new element.
HierName	The name of the hierarchy within the dimension.
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is empty, the new element is added to the end of the dimension.

Argument	Description
ElName	The name you want to assign to the new element.
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
HierarchyElementInsert('Region', 'Western', 'Belgium', 'Netherlands','N');
```

This example adds the numeric element Netherlands to the Western hierarchy in the Region dimension. Netherland displays immediately before Belgium in the dimension definition.

HierarchyElementInsertDirect

HierarchyElementInsertDirect adds an element to a dimension by directly editing the dimension. You can use this function to add numeric, string, or consolidated elements.

This function is valid in TM1 TurboIntegrator processes only.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like HierarchyElementInsert) are used in the metadata tab of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element insertion needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
HierarchyElementInsertDirect(DimName, HierName, InsertionPoint, ElName, ElType);
```

Argument	Description
DimName	The dimension to which you want to add a new element.
HierName	The name of the hierarchy within the dimension.

Argument	Description
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is empty, the new element is added to the end of the dimension. Note that this function is optimized for the case where the InsertionPoint is passed as an empty string.
ElName	The name you want to assign to the new element.
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
HierarchyElementInsertDirect('Region', 'Western', 'Belgium', 'Netherlands','N');
```

This example adds the numeric element Netherlands to the Western hierarchy in the Region dimension. Netherlands displays immediately before Belgium in the dimension definition.

HierarchyElementPrincipalName

HierarchyElementPrincipalName returns the principal name of an element or element alias.

TurboIntegrator must use principal element names when updating dimensions; element aliases cannot be used. This function is therefore useful for determining principal element names while attempting to update a dimension when only element aliases are available to the TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementPrincipalName( DimName, HierName, ElName )
```

Argument	Description
DimName	The name of the dimension from which you want to retrieve a principal element name.
HierName	The name of the hierarchy within the dimension.
ElName	An element name. ElName can be either an element alias or a principal element name.

Example

If ElName is not in the currently saved version of DimName, the function returns ElName.

If ElName is in DimName, whether as an element alias or a principal element name, it returns the principal name of the element.

HierarchyExists

HierarchyExists determines whether a specific hierarchy exists on the server from which a TurboIntegrator process is executed. The function returns 1 if the hierarchy exists on the server, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyExists(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension that contains the hierarchy whose existence you want to confirm.
HierName	The name of the hierarchy within the dimension.

Example

```
HierarchyExists('Region', 'Europe');
```

This example determines if the Europe hierarchy, in the Region dimension, exists on the server.

HierarchyHasOrphanedLeaves

HierarchyHasOrphanedLeaves returns 1 if there are one or more elements in the specified hierarchy that are not components of a parent element in that hierarchy.

The function returns 1 if the hierarchy exists on the server, otherwise it returns 0. This function returns 0 if all leaf elements in the hierarchy are a component of one or more parent elements. Values stored against such elements will not be aggregated.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyHasOrphanedLeaves(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension that contains the hierarchy being reviewed.
HierName	The name of the hierarchy you are reviewing for orphaned leaf members.

Example

```
HierarchyHasOrphanedLeaves('Region', 'Europe');
```

This example determines if the Europe hierarchy, in the Region dimension, contains any orphaned leaves.

HierarchySortOrder

HierarchySortOrder sets a sort type and sense for dimension elements and for components of consolidated elements within a dimension. The sort order defined by DimensionSortOrder determines how the subset displays in the Subset Editor.

DimensionSortOrder sets properties for a dimension; the dimension is not actually sorted until it is saved on the server.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySortOrder(DimName, HierName, CompSortType, CompSortSense,ElSortType , ElSortSense);
```

Argument	Description
DimName	The name of the dimension for which you want to set a sort order.
HierName	The name of the hierarchy within the dimension.
CompSortType	Defines how components of consolidated elements appear in the dimension. There are two CompSortType values: ByInput - Retains the order in which components were originally inserted into consolidations. ByName - Sorts components of consolidations by name.
CompSortSense	Defines the sort sense for components of consolidations. This is a required argument, but it applies only when the CompSortType is ByName. There are two possible CompSortSense values: Ascending - Sorts components of consolidations in ascending alphabetical order. Descending - Sorts components of consolidations in descending alphabetical order.
ElSortType	Defines a sort order for dimension elements. There are four possible ElSortType values: ByInput - Retains the order in which elements were originally inserted into the dimension. ByName - Sorts dimension elements by name. ByLevel - Sorts dimension elements by level. ByHierarchy - Sorts dimension elements by hierarchy.

Argument	Description
ElSortSense	Defines the sort sense for dimension elements. This is a required argument, but it applies only when the ElSortType is ByName or ByLevel. There are two possible ElSortSense values: Ascending - Sorts dimension elements in ascending order, either alphabetically or by level. Descending - Sorts dimension elements in descending order, either alphabetically or by level.

Example

```
HierarchySortOrder ('Region', 'Europe', 'ByName', 'Descending','ByLevel', 'Ascending');
```

This example sets a sort order for the Europe hierarchy in the Region dimension. All dimension elements are sorted by level in ascending order, and any components of consolidations are sorted in descending alphabetical order.

HierarchyTimeLastUpdated

HierarchyTimeLastUpdated indicates when a specified dimension hierarchy was last updated. The function returns a real number that represents the current day (including the hour, minute, second, and millisecond) since the beginning of the year 1900.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyTimeLastUpdated(dimension, hierarchy);
```

Argument	Description
dimension	The name of the dimension.
hierarchy	The name of the hierarchy.

Example

```
HierarchyTimeLastUpdated('Region', 'Europe');
```

This example returns information on when the Europe hierarchy of the Region dimension was last updated. If a value of 42548.<hours>.<minutes>.<milliseconds> is returned, you can divide 42548 by 365 to obtain (approximately) 116. When added to the started of 1900, the result is a current year of 2016.

HierarchyTopElementInsert

HierarchyTopElementInsert creates a root element in a dimension. If the dimension already has a single root, then this element will not be created.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyTopElementInsert(DimName, HierName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.
HierName	The name of the hierarchy within the dimension.
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
HierarchyTopElementInsert('Region', 'Western', 'Netherlands', 'World');
```

This example adds the root element World to the Western hierarchy of the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

HierarchyTopElementInsertDirect

HierarchyTopElementInsertDirect creates a root element in a dimension by directly editing the dimension. If the dimension already has a single root, then this element will not be created.

This function is valid in TM1 TurboIntegrator processes only.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like HierarchyTopElementInsert) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
HierarchyTopElementInsertDirect(DimName, HierName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.

Argument	Description
HierName	The name of the hierarchy within the dimension.
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
HierarchyTopElementInsertDirect('Region', 'Western', 'Netherlands', 'World');
```

This example adds the root element World to the Western hierarchy of the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

HierarchyUpdateDirect

HierarchyUpdateDirect performs a full rewrite of a hierarchy that has been subject to direct editing in a TurboIntegrator process, essentially compacting the memory footprint of the hierarchy.

A dimension that undergoes a series of direct-only edits (element deletions, in particular) will eventually use more memory than its fully-rewritten counterpart would. This function can optionally be used after directly editing a dimension with HierarchyElementInsertDirect, HierarchyElementDeleteDirect, HierarchyElementComponentAddDirect, HierarchyElementComponentDeleteDirect, and/or HierarchyTopElementInsertDirect. Calling HierarchyUpdateDirect incurs an initial full-copy memory cost, however it can be used to guarantee that the dimension is at its smallest possible memory footprint after processing is complete.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyUpdateDirect(DimName, HierName);
```

Argument	Description
DimName	The name of the dimension you want to rewrite.
HierName	The name of the hierarchy within the dimension.

Example

```
HierarchyUpdateDirect('Region', 'Western');
```

This example rewrites the Western hierarchy of the Region dimension.

ODBC TurboIntegrator Functions

These functions facilitate the ODBC manipulation.

ODBCClose

ODBCClose closes a connection to an ODBC data source.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ODBCClose(Source);
```

Argument	Description
Source	The name of an open ODBC data source.

Example

```
ODBCClose('Accounting');
```

This example closes the connection to the Accounting ODBC source.

ODBCOpen

ODBCOpen opens an ODBC data source for output.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ODBCOpen(Source, ClientName, Password);
```

Argument	Description
Source	An ODBC data source name.
ClientName	A valid client on the data source.
Password	A password for the ClientName.

Example

```
ODBCOpen('Accounting', 'Jdoe', 'Bstone');
```

This example opens the Accounting ODBC data source for the Jdoe client using the password Bstone.

ODBCOPENEx

ODBCOPENEx opens an ODBC data source for output specifying that the connection should be opened as a Unicode connection.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

Format is: ODBCOPENEx (dataset name, dataset client name, client password, (use-Unicode-interface flag))

```
ODBCOpenEx(Source, ClientName, Password, UseUnicodeODBC);
```

Argument	Description
Source	An ODBC data source name.
ClientName	A valid client on the data source.
Password	A password for the ClientName.

Argument	Description
UseUnicodeODBC	Defines the type of Unicode connection to use.

Example

```
ODBCOpenEx( TestTable, sa, , 1 );
```

```
chinese= ;
chinese = CHARW( 37123 );
fieldval = chinese | SomeNewText;
sql= Update TestTable set ForeName = N | fieldval |  WHERE CustomerId= 1
ODBCOUTPUT( Unicode, sql );
```

The result SQL statement looks like:

```
Update TestTable set ForeName = N?SomeNewText WHERE
CustomerId = 1
```

ODBCOutput

ODBCOutput executes an SQL update query against an open ODBC data source. You should use the ODBCOpen function to open the data source before calling ODBCOutput, and use ODBCClose to close the data source before exiting the process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ODBCOutput(Source, SQLQuery, [SQLQuery2, SQLQuery3, ...]);
```

Argument	Description
Source	The ODBC data source against which you want to run a query.
SQLQuery	An SQL query statement. Though ODBCOutput was developed to update tables, it can be used to execute any SQL query on the data source. In circumstances where the SQL query statement exceeds 255 characters, you should split the query into multiple SQLQuery arguments (SQLQuery2, SQLQuery3, etc.). This lets you create query statements that exceed the 255 character limit for TurboIntegrator arguments. When the ODBCOutput function is executed, all SQLQuery arguments are concatenated and the query is successfully executed.

Example

```
ODBCOutput('Accounting', 'INSERT [CategoryID], [CategoryName]FROM Categories;');
```

This example executes the specified query against the Accounting data source.

SetODBCUnicodeInterface

SetODBCUnicodeInterface sets whether the ODBC interface should use the Unicode wide functions or the regular single-byte character functions. Setting this function to 1 uses the wide character ODBC interface.

Some ODBC driver support either the older single-byte interface as well as a Unicode style 'wide-character' interface, where characters are passed and retrieved as 16-bit quantities. If the driver chosen does not support one or the other style, a flag is provided to force TurboIntegrator to use a particular style of interface.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetODBCUnicodeInterface=1
```

Argument	Description
1	Use the wide character ODBC interface.
0	Use the single-byte interface.

Process Control TurboIntegrator Functions

These functions pertain to process control.

ExecuteCommand

ExecuteCommand executes a command line during a process. You can use ExecuteCommand to run a desktop application, but not a service.

If you use ExecuteCommand to run an executable, the following conditions apply:

- If the CommandLine argument specifies only the name of a file to be executed, a Windows server looks for the file in both the server database directory and in the directory where Tm1s.exe resides. A UNIX server looks for the file only in the server database directory.
- If the CommandLine argument uses a relative path prefix, both the Windows and UNIX server attempt to locate the file in the server database directory only.
- On either the Microsoft Windows or UNIX server, you can pass an absolute path to the CommandLine argument to execute a file in any location..

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ExecuteCommand(CommandLine, Wait);
```

Argument	Description
CommandLine	The command line you want to execute.
Wait	Indicates if the process should wait for the command to complete execution before continuing to the next process statement. An argument value of 0 causes the process to proceed to the next statement without waiting for the command line to execute. An argument value of 1 causes the process to wait for the command line to successfully execute before proceeding to the next statement.

ExecuteProcess

ExecuteProcess lets you execute a TurboIntegrator process from within another process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ExecuteProcess(ProcessName, [ParamName1, ParamValue1,ParamName2, ParamValue2]);
```

Argument	Description
ProcessName	The name of the process to be executed. This process must reside on the same server as the process from which ExecuteProcess is called. If the process named by this argument cannot be found at runtime, the calling process is immediately terminated. (TurboIntegrator does not check for a valid ProcessName at compilation.)
ParamName	The name of an existing parameter of the process to be executed. This argument is required only if the process to be executed uses parameters.
ParamValue	A valid value for the ParamName parameter. If you specify a ParamName argument, you must specify a corresponding ParamValue. The ParamName and ParamValue arguments must occur in ordered pairs, with the name of the parameter followed by the value. You must specify a ParamName and corresponding ParamValue for each parameter of the process to be executed.

The parameter names passed in the ExecuteProcess function are matched at runtime against the parameter names specified in the process to be executed. If the passed names cannot be found in the parameter list of the process to be executed, a serious error results, causing the immediate termination of the process from which ExecuteProcess is called.

Return Values

ExecuteProcess returns a real value that can be tested against one of the following return value functions:

Function	Description
ProcessExitByChoreQuit()	indicates that the process exited due to execution of the ChoreQuit function
ProcessExitNormal()	indicates that the process executed normally
ProcessExitMinorError()	indicates that the process executed successfully but encountered minor errors
ProcessExitByQuit()	indicates that the process exited because of an explicit "quit" command
ProcessExitWithMessage()	indicates that the process exited normally, with a message written to `tm1server.log`
ProcessExitSeriousError()	indicates that the process exited because of a serious error

Function	Description
ProcessExitOnInit()	indicates that the process aborted during initialization
ProcessExitByBreak()	indicates that the process exited because it encountered a ProcessBreak function

Example

To record when a process called by ExecuteProcess fails because of a serious error, use code similar to the following:

```
return_value = ExecuteProcess('create_sales_cube');
ASCIIOutput('C:\temp\process_return_value.txt', 'Process exited
with serious errors at', TIME, 'on', TODAY);if(return_value = ProcessExitSeriousError() )
endif;
```

GetProcessErrorFileDirectory

GetProcessErrorFileDirectory returns the full pathname, with trailing slash, of the directory where TurboIntegrator process error files are written. By default, all process error log files are written to the data directory of the server on which the process resides.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
GetProcessErrorFileDirectory;
```

Arguments

None.

GetProcessErrorFilename

GetProcessErrorFilename returns the name of the TurboIntegrator process error log file associated with a process. If the process has not yet generated an error log file, the function returns an empty (null) string.

Important: A process error log file is not generated until all statements in a given process tab (Prolog, Metadata, Data, or Epilog) have executed. Accordingly, you can use GetProcessErrorFilename to check if any previous tabs have generated an error log file, but you cannot use the function to determine if the current process tab causes errors to be written to a log file.

For example, by determining that GetProcessErrorFilename returns a non-null string in the Epilog tab, you can tell that errors were generated in the Prolog, Metadata, or Data tabs. However, you cannot use GetProcessErrorFilename in the Data tab to determine if the Data tab generates errors.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
GetProcessErrorFilename;
```

Arguments

None.

GetProcessName

GetProcessName returns as a string the name of the current process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
GetProcessName()
```

Arguments

None.

```
Name = GetProcessName();
```

If

The If statement allows a process to execute a statement or series of statements when a given expression is true. You can use arithmetic operators, logical operators, and comparison operators to construct an expression.

The TurboIntegrator If statement differs from the Rules IF function in that the TurboIntegrator statement can accept multiple ElseIf or Else statements to evaluate multiple expressions, while the Rules IF function can evaluate only one expression.

You can nest up to 20 If/ElseIf/Else statements in a TurboIntegrator process. If you exceed 20 nested If/ElseIf/Else statements, you will receive an error when attempting to save the process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
If(expression);
statement1;
ElseIf(expression);
statement2;
ElseIf(expression);
statement3;
Else;
statement4;
EndIf;
```

Arguments

None.

Examples

```
If (x=5);
    ASCIIOutput('c:\temp\if.txt','x equals five');
ElseIf (x=1);
    ASCIIOutput ('c:\temp\if.txt', 'x equals one');
ElseIf (x=2);
    ASCIIOutput ('c:\temp\if.txt', 'x equals two');
ElseIf (x=3);
    ASCIIOutput ('c:\temp\if.txt', 'x equals three');
ElseIf (x=4);
    ASCIIOutput ('c:\temp\if.txt', 'x equals four');
Else;
    ASCIIOutput ('c:\temp\if.txt', 'x falls outside expected range');
EndIf;
```

This example evaluates the value of X. If X=5, the ASCIIOutput function is executed to write the string "x equals five" to c:\temp\if.txt. If X does not equal 5, the first ElseIf statement is evaluated. If X=1, the ASCIIOutput function is executed to write the string "x equals one" to c:\temp\if.txt. This processing continues until the EndIf is executed.

Simple If statements can also be constructed without the use of ElseIf, as in this example:

```
IF(expression);
    statement1;
ELSE;
```

```
    statement2;
ENDIF;
```

ItemReject

ItemReject rejects a source record and places it in the error log, along with a specified error message.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ItemReject(ErrorString);
```

Argument	Description
ErrorString	The error message you want written to the error log when a record is rejected.

Example

```
ItemReject(' Value outside of acceptable range.');
```

This example places a source record in the error log, along with the error message 'Value outside of acceptable range.' when the source record contains a value that is beyond a defined range.

ItemSkip

ItemSkip forces a process to skip the current data source item.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ItemSkip;
```

Arguments

None.

ProcessBreak

ProcessBreak stops processing source data and proceeds to the Epilog portion of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessBreak;
```

Arguments

None.

ProcessError

ProcessError causes an immediate termination of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessError;
```

Arguments

None.

ProcessExists

ProcessExists determines whether a specific TurboIntegrator process exists.

The ProcessExists function returns one of three possible values:

- If a TurboIntegrator process with the specified name does not exist, the function returns 0.
- If a process with the specified name does exist and is valid, the function returns 1.
- If a process with the specified name does exist, but has compilation errors, the function returns -1.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessExists(ProcessName);
```

Argument	Description
ProcessName	The name of the process for which you are trying to determine status.

ProcessExitByChoreRollback

ProcessExitByChoreRollback initiates a chore rollback and exits with an error code. Similar to ChoreRollback, when used inside a TurboIntegrator process, this function throws out all pending edits and cancels further processing. An error message appears in the tm1server.log and tm1processorerrorXXX.log files.

When used in a single-commit mode chore, ProcessExitByChoreRollback throws out all pending edits from all previous processes and exits.

When used in a multi-commit mode chore, ProcessExitByChoreRollback throws out all pending edits from the current processes and then exits. Changes that have already been committed cannot be rolled back.

ProcessExitByChoreRollback returns the error code number.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessExitByChoreRollback;
```

Arguments

None.

ProcessExitByProcessRollback

ProcessExitByProcessRollback initiates a process rollback and exits with an error code. Similar to ProcessRollback, when used inside a TurboIntegrator process, this function throws out all pending edits and cancels further processing. An error message appears in the tm1server.log and tm1processorerrorXXX.log files.

When used in a single-commit mode chore, ProcessExitByProcessRollback throws out all pending edits from all previous processes and exits.

When used in a multi-commit mode chore, ProcessExitByProcessRollback throws out all pending edits from the current process and then exits. Changes that have already been committed cannot be rolled back.

ProcessExitByProcessRollback returns the error code number.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessExitByProcessRollback;
```

Arguments

None.

ProcessQuit

ProcessQuit terminates a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessQuit;
```

Arguments

None.

ProcessRollback

ProcessRollback initiates a process rollback. When used inside a TurboIntegrator process, this function throws out all pending edits and cancels further processing. An error message appears in the tm1server.log and tm1processorerrorXXX.log files.

Note: In IBM Planning Analytics version 2.0.8 or later, when a TurboIntegrator process rolls back and restarts, the process is now represented in the `tm1server.log` file as three steps: starting, restarting because of lock contention or rollback, and then finishing.

An entry is added to the `tm1server.log` file that shows the TurboIntegrator process as restarting due to lock contention or rollback instead of just starting. This logging is enabled by default without setting any specific debug options.

When used in a single-commit mode chore, ProcessRollback throws out all pending edits from all previous processes and continues execution at the next process in the chore. If lock contention is encountered after the call to ProcessRollback, the entire chore is restarted.

When used in a multi-commit mode chore, ProcessRollback throws out all pending edits from the current process and then continues execution at the next process in the chore. Changes that have already been committed cannot be rolled back. If lock contention is encountered after the call to ProcessRollback, only the current process is restarted.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ProcessRollback;
```

Arguments

None.

RunProcess

RunProcess lets you run TurboIntegrator processes in parallel, each on its own thread that is managed by TM1 Server. This approach speeds up data load and other operations where TurboIntegrator processes are used to divide the work.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RunProcess(ProcessName, [ParamName1, ParamValue1,ParamName2, ParamValue2]);
```

Argument	Description
ProcessName	The name of the process to be run. This process must reside on the same server as the process from which RunProcess is called. If the process named by this argument cannot be found at runtime, the calling process is immediately terminated. (TurboIntegrator does not check for a valid ProcessName at compilation.)
ParamName	The name of an existing parameter of the process to be run. This argument is required only if the process to be run uses parameters.
ParamValue	A valid value for the ParamName parameter. If you specify a ParamName argument, you must specify a corresponding ParamValue. The ParamName and ParamValue arguments must occur in ordered pairs, with the name of the parameter followed by the value. You must specify a ParamName and corresponding ParamValue for each parameter of the process to be run.

The parameter names passed in the RunProcess function are matched at runtime against the parameter names specified in the process to be run. If the passed names cannot be found in the parameter list of the process to be run, a serious error results, causing the immediate termination of the process from which RunProcess is called.

Return values

RunProcess returns a string. The string is the JobID, or an empty string if an error occurs.

Synchronized

Synchronized is used in a TurboIntegrator script to force serial execution of a designated set of TurboIntegrator processes.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

The synchronized() function uses the following syntax.

```
synchronized (lockName);
```

Synchronized takes a single required parameter that is a user-defined name for a lock object. This lock object name can be used in multiple TurboIntegrator processes in order to serialize their execution as a group.

Argument	Description
lockName	The user-defined name of a lock object on which to synchronize. Names are case-insensitive and embedded spaces are ignored. Names may not exceed 1023 characters in length. String/Yes/None

Semantics

A TurboIntegrator process may make any number of calls to synchronized(), with any number of lock objects. Serializing is effective from the time synchronized() is called, until the containing transaction completes.

For example, if synchronized() is called from a subprocess (Ps) of master process (Pm) or master chore (Cm), the Lock Object is "released" when Pm or Cm completes. The exception is that a SaveDataAll (SDA) prematurely "ends" a transaction mid-process execution; this applies to Lock Objects as well.

The synchronized() call may be placed anywhere within a TurboIntegrator script, but serialization applies to the entire TurboIntegrator process when it is encountered.

Consider a TurboIntegrator process with a synchronized() call somewhere in the "middle" of its script, and an operation O1 preceding that call. Two instances of this TurboIntegrator process may start at the same time. It is possible for one instance to run to completion, including its call to synchronized(), before the second instance reaches its synchronized() call. In this case, the two processes appear to the user to have run concurrently. If, instead, the second process does reach its synchronized() call before the first completes, it will undo any work it had done (O1) and wait for the first to complete. In this case, the two processes appear to the user to have serialized.

To avoid such confusion, and to optimize the use of synchronized(), it is recommended (but not enforced) that synchronized() calls be the first statements of a TurboIntegrator process.

Example

Consider that TurboIntegrator process P needs to update two cubes, Cube_1 and Cube_2.

Other TurboIntegrator processes may also need to update Cube_1 or Cube_2.

To cause all TurboIntegrator processes that will update Cube_1 or Cube_2, to run one at a time, P could call synchronized() in the following way:

```
sCube_1='Cube_1';
sCube_2='Cube_2';
sE1='Elm1';
sE2='Elm2';
sE4='Units';
sE5='Price';

Synchronized( sCube_1 );
Synchronized( sCube_2 );

CellPutn( 111, sCube_1, sE1, sE2 );
CellPutn( 9.99, sCube_2, sE4, sE5 );

# ...
```

Other TurboIntegrator processes that will update Cube_1 or Cube_2 must also call `synchronized( sCube_1 )` and/or `synchronized( sCube_2 )` in a similar way.

In this example, the two lock objects' names were chosen to be the same as the cubes' names. But a lock object's name does not have to be the same as other TM1 objects (cubes, dimensions, subsets).

While

The While statement allows a process to repeat a series of statements while a given condition is true. While statements can be nested.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
WHILE(logical expression);

statement1;

statement2;
```

```
...
statement n;
END;
```

Note: All WHILE statements must conclude with an END statement.

Arguments

None.

Rules Management TurboIntegrator Functions

These functions facilitate rules management.

CubeProcessFeeders

CubeProcessFeeders reprocesses all feeders in the rules for a specified cube.

This function reprocesses all feeders in the rules for a specified cube. The feeders are normally reprocess automatically when a rule file edit is saved, however, if the data changes, and those data changes will change some conditional feeders, this function will need to be called to get those conditional feeders re-evaluated.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeProcessFeeders(CubeName);
```

Argument	Description
CubeName	The cube for which you want to reprocess feeders.

Example

```
CubeProcessFeeders('2003sales');
```

This example reprocesses all feeders in the rules for the 2003sales cube.

CubeRuleAppend

CubeRuleAppend appends a single line of rule text to a TM1 cube rule.

Essentially, this function adds a single line of text to a rule (.rux) file. The line of text is typically a rule statement, but can also be a comment. If there is no rule associated with the cube at the time this function is executed, a new rule will be created, containing only the passed line.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeRuleAppend(CubeName, RuleText, IsCalculationRule);
```

Argument	Description
CubeName	The name of the cube associated with the rule to which you want to append a line of text.

Argument	Description

RuleText	The single line of text you want to append to the rule.
	The entire line of text you add must be enclosed in single quotes and must adhere to rules syntax conventions.
	If the line of text includes any element references, the element names must be enclosed in double single quotes to escape the single quotes that normally enclose element names. For example, a reference to an element named CL3 must be specified as `[''CL3'']`.
	The following are examples of valid lines of text you might append to a rule:
	`'[''CL3''] = [''CL4''] + [''Trial''];'`
	`'skipcheck;'`
	`'[''Trial''] => [''CL3''];'`

IsCalculationRule	The IsCalculationRule parameter declares whether the line should be inserted just above any feeder section that might exist in the cube rule. If the IsCalculationRule parameter is omitted, or passed as 0.0, then the new line will simply be appended to the end of the rule. Because rule (.rux) files consist of a calculation section followed by an optional feeder section, any appended lines that are calculation rule statements (or corresponding comments) should use a 1.0 for this argument to ensure that the new line is inserted in at the appropriate location in the rule file.

Examples

```
CubeRuleAppend( 'MyCube', '[''CL3''] = [''CL4''] + [''Trial''];', 1.0 );
```

This example inserts the calculation statement `['CL3'] = ['CL4'] + ['Trial'];` at the end of the calculation section of the rule for the MyCube cube.

```
CubeRuleAppend( 'MyCube', '[''Trial''] => [''CL3''];', 0.0 );
```

This example inserts the feeder statement `['Trial'] => ['CL3'];` at the end of the rule for the MyCube cube.

CubeRuleDestroy

CubeRuleDestroy deletes any rule that exists for a specified cube.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CubeRuleDestroy(CubeName);
```

Argument	Description
CubeName	The name of the cube associated with the rule that you want to delete

Example

```
CubeRuleDestroy('SalesProjections');
```

This example deletes the rule for the SalesProjectionscube.

DeleteAllPersistentFeeders

DeleteAllPersistentFeeders deletes any .feeder files that have persisted. When this function is used, all cubes are marked as "do not save feeders" so a subsequent SaveData will not persist feeders which means all feeders will be re-calculated on a server re-start.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DeleteAllPersistentFeeders;
```

Arguments

None.

ForceSkipCheck

ForceSkipCheck forces the query to perform as if the cube had a SKIPCHECK in the rules.

This function is valid in TM1 TurboIntegrator processes only.

This means that the query will process only values actually in the cube, as opposed to (the no SKIPCHECK case) where every possible cell would be enumerated looking for values. This function sets the state of the view query to select only values in the cube. The function must be added to the Prolog section of the TurboIntegrator process. By placing the ForceSkipCheck() in the Prolog it effects the entire view query of data elements to follow.

Syntax

```
ForceSkipCheck()
```

Arguments

None.

RuleLoadFromFile

RuleLoadFromFile creates a TM1 rule for a specified cube from a text file. Each rule statement must end with a semi-colon (;) and comments must be prefixed with the # character. If a rule already exists for the specified cube, the rule is overwritten by the rule created by RuleLoadFromFile.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RuleLoadFromFile(Cube, TextFile);
```

Argument	Description
Cube	The name of the cube for which you want to create a rule.
TextFile	The name of the text file from which you want to create a rule. You can specify the full path to this file, including file name and extension. (Example 1 below.) If you specify only the file name and extension, TurboIntegrator looks for the file in the server's data directory. If you do not specify a file extension, TurboIntegrator assumes the .rux extension by default. (Example 2 below.)

If you leave the TextFile argument empty, TurboIntegrator looks for a source file with the same name as the cube (but with a .rux extension) in the server's data directory. (Example 3 below.)

Example

```
RuleLoadFromFile('Sales', 'C:\temp\cuberule.txt');
```

This example uses the contents of the cuberule.txt file in the C:\temp directory to create a rule for the Sales cube.

```
RuleLoadFromFile('Sales', 'cuberule');
```

This example creates a rule for the Sales cube using the file named cuberule.rux in the server's data directory.

```
RuleLoadFromFile('Sales', ' ');
```

This example creates a rule for the Sales cube using the file named Sales.rux in the server's data directory.

Sandbox Functions

These functions are used with sandboxes.

GetUseActiveSandboxProperty

GetUseActiveSandboxProperty returns a Boolean value that indicates whether a process reads and writes data to the base data or to the user's active sandbox.

This function is valid in TM1 TurboIntegrator processes only.

The default is for processes to read and write to the base data.

- If the return is 0, the process is currently reading and writing to the base data.
- If the return is 1, the process is currently reading and writing to the active sandbox.

Note: This function returns the permanent value for this property as set in the Architect / Server Explorer user interface *unless*you have used the SetUseActiveSandboxProperty function in the process. In that case, the value for this property is determined by the value that was last set with the SetUseActiveSandboxProperty function.

Syntax

```
GetUseActiveSandboxProperty()
```

Arguments

None.

Example

```
return_value = GetUseActiveSandboxProperty();
```

This example will return a Boolean value indicating whether the process is currently reading and writing cube data to the active sandbox or to the base data.

ServerActiveSandboxGet

ServerActiveSandboxGet returns the name of the user's active sandbox. If the user has no active sandbox, an empty string is returned. Because chores run in the context of a special admin user, and can have no active sandbox, this function always returns an empty string when executed using a chore.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerActiveSandboxGet();
```

Arguments

None.

Example

```
return_value = ServerActiveSandboxGet();
```

This example will return the active sandbox of the user executing the TI process in which the function call is made.

ServerActiveSandboxSet

ServerActiveSandboxSet sets the active sandbox of the executing user. An empty string is used to clear the executing user's active sandbox. This function throws an error if the executing user does not own a sandbox with the passed name.

Because chores run in the context of a special admin user, and can have no active sandbox, this function always throws an error when executed using a chore.

Note: For a TurboIntegrator process to read and write values in the context of the executing user's active sandbox, the UseActiveSandbox property must be set. See "GetUseActiveSandboxProperty" on page 549 and "SetUseActiveSandboxProperty" on page 558.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerActiveSandboxSet(SandboxName)
```

Argument	Description
SandboxName	A string value. The name of a sandbox owned by the executing user.

Example: Set the executing user's active sandbox to "Best case"

```
ServerActiveSandboxSet('Best case');
```

Example: Clear the executing user's active sandbox and set context back to the base data

```
ServerActiveSandboxSet('');
```

ServerSandboxClone

ServerSandboxClone clones an existing sandbox into a new sandbox.

Sandboxes are private workspaces in which a user can enter and store data values separate from TM1 base data. Sandboxes are stored on disk and, when in use, in memory.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxClone(sandboxName,newSandboxName );
```

Argument	Description
sandboxName	A string value. The name of a sandbox owned by the executing user.
newSandboxName	A string value. The name of a sandbox to be created as a clone of *sandboxName*.

Example

```
ServerSandboxClone( 'Best case', 'Second best case');
```

ServerSandboxCreate

ServerSandboxCreate creates a new sandbox.

Sandboxes are private workspaces in which a user can enter and store data values separate from TM1 base data. Sandboxes are stored on disk and, when in use, in memory.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxCreate( sandboxName );
```

Argument	Description
sandboxName	A string value. The name of a sandbox to be created.

Example

```
ServerSandboxCreate( 'My sandbox' );
```

ServerSandboxesDelete

ServerSandboxesDelete allows administrators to discard user sandboxes that match certain criteria.

Sandboxes are private workspaces in which a user can enter and store data values separate from TM1 base data. Sandboxes are stored on disk and, when in use, in memory.

This function operates server side and is available through TurboIntegrator and the API function ServerSandboxesDelete. Using this feature in a TurboIntegrator process, administrators can schedule maintenance using automated chores.

This function is valid in TM1 TurboIntegrator processes only.

Description

This function uses a "predicate" to describe the sandbox being deleted. A predicate can be read as: "Delete sandboxes whose *attribute* is *condition value*."

For example: "Delete sandboxes whose size is greater than 10 MB." In this example, the attribute is the "size" of the sandbox, the condition is "greater than", and the value is "10 MB".

There are two optional delimiter character parameters to the TurboIntegrator function. Because a sandbox has no restrictions on which characters can be used in their name, administrators can supply their own "safe" delimiter when needed.

For example, `ServerSandboxesDelete( 'client:=:Admin, name:=:best case scenario' );"`

In the following example, the colon character is used in the sandbox name ("best::case::scenario") so another delimiter is needed:

```
ServerSandboxesDelete( 'client|=|Admin# name|=|best::case::scenario', '|', '#' );"
```

Note: The exact syntax of a predicate differs between the TurbIntegrator and API forms of this function.

Syntax

```
ServerSandboxesDelete(string,string,string)
```

Argument	Description
Predicates	The name of the process to be executed. This process must reside on the same server as the process from which RunProcess is called. Required String No default An arbitrary length list of predicates. Each predicate is a string containing three tokens. The first token indicates an attribute of a sandbox. The second indicates a condition, for example ">" or "=". The third token is a possible value of the attribute on which sandboxes should be conditionally filtered. The entire string may not exceed 10,000 characters in length.
PredicateDelimiter	Optional String default is : (colon) Optional delimiter character. The string may not exceed 1 character in length.

Argument	Description
PredicateListDelimiter	Optional String default is , (comma) Optional delimiter character. The string may not exceed 1 character in length.

Filter Attributes

Filter attributes are properties of a sandbox on which it can be conditionally matched. Attribute names and their corresponding valid conditions are case insensitive and ignore embedded whitespace. For example, the following two calls are both valid:

```
ServerSandboxesDelete( 'client:=:Admin' );
```

```
ServerSandboxesDelete( 'C L I E N T : = :Admin' );
```

Table 9. Filter Attributes

Attribute	Description	Valid Conditions	Value Type
UpdateDate	Timestamp of the last write action performed in the sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
AccessDate	Timestamp of the last unload of a sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
CreationDate	Timestamp of the creation of a sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
Size	The in-memory size of a sandbox.	<, =, >.	Size following log4cxx's conversion rules (see configuration parameter AuditLogMaxTemp FileSize) For example, 10 MB. Kilobytes are the most granular units.
Name	The name of a sandbox.	=, containing.	String.
Client	The owning client of a sandbox.	=.	String.
Group	A group of which the owning client of a sandbox is a member.	=.	String.

Logging and Returns

Sandbox deletion is logged using the preexisting audit logging functionality. Additionally, a more detailed report of the effects of sandbox administration is included in the debug log (tm1server.log) at INFO level. This report will include the list of affected sandboxes, as well as some of their attributes, and any errors encountered.

ServerSandboxesDelete returns only a success or failure status.

Semantics

Predicate List

Multiple predicates passed in a single call to ServerSandboxesDelete are conjunctive. In other words, for a sandbox to match the passed criteria, all predicates must be true. Multiple calls to ServerSandboxesDelete can be used to achieve disjunctive behavior. Only one occurrence of each attribute is allowed per call to ServerSandboxesDelete. For example, passing client twice is invalid as a sandbox has only one owning client. When multiple occurrences of an attribute are detected, a warning displays in the detailed report, however, the operation will not abort in failure. In such a case, the predicates are tested as with any other query, but the results set is always empty.

Locking

To avoid massive locking issues, ServerSandboxesDelete looks at the sandboxes of a client as a point-in-time snapshot and then, when possible, release any locks that would ensure a serializable transaction. Because of this behavior, once a client is "passed" in the iteration of all clients, a sandbox matching the filter criteria may be added to that client before the maintenance transaction completes. This behavior is similar to the behavior that occurs when a sandbox is added to the client immediately after the transaction completes.

Scope

Members of the ADMIN (super-user) and the DataAdmin groups will have access to all sandboxes of all clients. They must explicitly specify the client attribute to limit the scope of their call to ServerSandboxesDelete to only their own sandboxes. All other users have access to only their own sandboxes; if they specify a different client, or a group to which they do not belong, the function will abort in failure and return a privilege error.

In-Use Sandboxes

When a sandbox meets the criteria for deletion, but is currently in use, that sandbox will not be deleted. An entry will appear in the debug log info-level report indicating the occurrence.

Access and Update Dates

Date attributes can be matches with, at most, day granularity. Because of this restriction, recording of these attributes is correspondingly granular. Last Update Date is not updated on individual cell writes. Instead, the system records the unload date of a sandbox that has had something written to it while it was loaded in memory. For such sandboxes, Last Access Date and Last Update Date will be the same. Only Last Access Date is updated on the unloading of a sandbox from memory. Also, because in-memory sandboxes are not subject to ServerSandboxesDelete, Last Access Date is not updated when a sandbox is loaded into memory.

For example, consider the follow usage scenario:

Table 10. Last Access Day Example

Day	Time	Action
1	1	Load Sandbox S
1	2	Write 1
2	3	Read 1
2	4	Unload Sandbox

A user is working with sandbox over the course of two days (perhaps for a much shorter period encompassing the day change.) At time 4, when the sandbox is unloaded, Last Update Date is set to 2, rather than 1 where the last update actually occurred. Last Access Date is also set to 2 at time 4 in this case. If Write1 were instead a read, only Last Access Date would be set to 2, while Last Update Date wouldn't be changed.

Example

```
ServerSandboxesDelete( 'client:=:Admin, name:=:best case scenario' );
```

ServerSandboxDiscardAllChanges

ServerSandboxDiscardAllChanges discards all changes in an existing sandbox.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxDiscardAllChanges( sandboxName );
```

Argument	Description
sandboxName	A string value. The name of a sandbox owned by the executing user.

Example

```
ServerSandboxDiscardAllChanges( 'MySandbox' );
```

ServerSandboxMerge

ServerSandboxMerge merges a source sandbox into an existing target sandbox. If the target sandbox is not specified, the source sandbox is merged into base.

Sandboxes are private workspaces in which a user can enter and store data values separate from TM1 base data. Sandboxes are stored on disk and, when in use, in memory.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxMerge( src, tgt, conflictRes, waitForLocks);
```

Argument	Description
src	The name of the source sandbox owned by the executing user to be merged with the *<tgt>* sandbox. The *<src>* sandbox is not changed. Required String
tgt	The name of a sandbox owned by the executing user to be merged with the *<src>* sandbox. The *<tgt>* sandbox is updated. If *<tgt>* is blank, you are merging *<src>* with base data and updating base. Required. To leave this parameter blank, use 2 concatenated single quotes: `''`. String

Argument	Description
conflictRes	The *<conflictRes>* parameter is ignored. Optional Numeric
waitForLocks	The *<waitForlocks>* parameter is an integer that indicates whether to wait for locks to guarantee serialization. 1 means wait for locks, catch any conflict exceptions, and retry instead of allowing the chore or process to roll back. 0 means do not wait for locks. Allow exceptions that cause rollback. Optional Numeric

Example

Merge `mySandbox` to base.

```
ServerSandboxMerge(mySandbox, '');
```

ServerSandboxExists

ServerSandboxExists tests for the existence of the passed sandbox. 1 is returned when the passed sandbox exists, 0 otherwise.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxExists( sandboxname )
```

or

```
ServerSandboxExists( sandboxname , username )
```

Arguments

The name of the sandbox whose existence is being tested. `ServerSandboxExists` takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

The following snippet shows how the `ServerSandboxExists`, `ServerSandboxGet`, and `ServerSandboxListCountGet` functions can be used to iterate the sandboxes of user called User1 and output those sandboxes to a text file. The TurboIntegrator process would successfully execute for members of the Admin or Data Admin groups and for user called User1. The TurboIntegrator process would fail with a privilege error for any other users.

```
SandboxIndex = 1;
NumSandboxes = ServerSandboxListCountGet( 'User1' );

WHILE( SandboxIndex <= NumSandboxes );
```

```
   SandboxName = ServerSandboxGet( SandboxIndex, 'User1' );

   IF( ServerSandboxExists( SandboxName, 'User1' ) = 1 );

      ASCIIOUTPUT( 'C:\User1Sandboxes.txt', SandboxName );

   ENDIF;

   SandboxIndex = SandboxIndex + 1;

END;
```

ServerSandboxGet

ServerSandboxGet returns the name of the sandbox identified by the number *N*, where *N* is the parameter entered.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxGet( index )
```

or

```
ServerSandboxGet( index, username)
```

Arguments

The index of the requested sandbox in the user's sandbox collection. The index space will be contiguous, so the first occurrence of an empty string return can be used to break iteration. Also, deleting a sandbox will alter the indexes of any sandboxes that follow that sandbox in the list.

`ServerSandboxGet` takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

The following snippet shows how the `ServerSandboxExists`, `ServerSandboxGet`, and `ServerSandboxListCountGet` functions can be used to iterate the sandboxes of user called User1 and output those sandboxes to a text file. The TurboIntegrator process would successfully execute for members of the Admin or Data Admin groups and for user called User1. The TurboIntegrator process would fail with a privilege error for any other users.

```
SandboxIndex = 1;
NumSandboxes = ServerSandboxListCountGet( 'User1' );

WHILE( SandboxIndex <= NumSandboxes );

   SandboxName = ServerSandboxGet( SandboxIndex, 'User1' );

   IF( ServerSandboxExists( SandboxName, 'User1' ) = 1 );

      ASCIIOUTPUT( 'C:\User1Sandboxes.txt', SandboxName );

   ENDIF;

   SandboxIndex = SandboxIndex + 1;

END;
```

ServerSandboxListCountGet

ServerSandboxListCountGet returns the count of sandboxes as a number.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerSandboxListCountGet()
```

or

```
ServerSandboxListCountGet( username )
```

Arguments

ServerSandboxListCountGet takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

The following snippet shows how the ServerSandboxExists, ServerSandboxGet, and ServerSandboxListCountGet functions can be used to iterate the sandboxes of user called User1 and output those sandboxes to a text file. The TurboIntegrator process would successfully execute for members of the Admin or Data Admin groups and for user called User1. The TurboIntegrator process would fail with a privilege error for any other users.

```
SandboxIndex = 1;
NumSandboxes = ServerSandboxListCountGet( 'User1' );

WHILE( SandboxIndex <= NumSandboxes );

   SandboxName = ServerSandboxGet( SandboxIndex, 'User1' );

   IF( ServerSandboxExists( SandboxName, 'User1' ) = 1 );

      ASCIIOUTPUT( 'C:\User1Sandboxes.txt', SandboxName );

   ENDIF;

   SandboxIndex = SandboxIndex + 1;

END;
```

SetUseActiveSandboxProperty

SetUseActiveSandboxProperty controls whether a process reads and writes cube data to the base data or to the user's active sandbox. The default is for processes to read and write to the base data.

The scope of this function applies only to the current running process and temporarily overrides the permanent value for this property that is set in the Architect / Server Explorer user interface.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetUseActiveSandboxProperty(PropertyValue)
```

Argument	Description
PropertyValue	A Boolean value that indicates whether the process should use the active sandbox context when reading and writing cube data. If PropertyValue = 0, the process will disregard the active sandbox context and read/write to the base data. If PropertyValue = 1, the process will read/write cube data to the active sandbox.

Example

```
SetUseActiveSandboxProperty(1);
```

This example will cause the process to read/write cube data to the active sandbox for the rest of this execution.

Security TurboIntegrator Functions

These functions pertain to security.

AddClient

AddClient creates a new client on the server. Changes applied through the AddClient functions do not take effect until the Metadata procedure in a process is completed. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AddClient(ClientName);
```

Argument	Description
ClientName	The name of the client you want to add to the server. The client name is limited to 255 characters/bytes.

Example

```
AddClient('Brian');
```

This example adds the client Brian to the server.

AddGroup

AddGroup creates a new user group on the server. Changes applied through the AddGroup function do not take effect until the Metadata procedure in a process is completed. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AddGroup(GroupName);
```

Argument	Description
GroupName	The name of the group you want to create.

Example

```
AddGroup('Finance');
```

This function adds the Finance user group to the server.

AssignClientToGroup

AssignClientToGroup assigns an existing client on a server to an existing user group. This function assigns an existing client on a server to an existing user group.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AssignClientToGroup(ClientName, GroupName);
```

Argument	Description
ClientName	The name of the client you want to assign to a group.
GroupName	The group to which you want to assign the client.

Example

```
AssignClientToGroup('Brian', 'Finance');
```

This example assigns the existing client Brian to the existing user group Finance.

AssignClientPassword

AssignClientPassword assigns a password to an existing client on a server. AssignClientPassword returns 1 if the password assignment is successful and returns 0 if the assignment fails.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AssignClientPassword (ClientName, Password);
```

Argument	Description
ClientName	The name of the client for which you want to assign a password.
Password	The password you want to assign to the client. When assigning a password, use plain text. TM1 will encrypt the password on the server. Passwords must be at least five characters in length.

Example

```
AssignClientPassword ('Brian', 'flyfisher');
```

This example assigns the password 'flyfisher' to the client named Brian.

AssociateCAMIDToGroup

AssociateCAMIDToGroup creates an association between a TM1 user group and a CAMID.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AssociateCAMIDToGroup(GroupName, CAMID, CAMIDDisplayValue);
```

Argument	Description
GroupName	The name of the TM1 group you want to associate with the CAMID.
CAMID	The name of the CAMID group. If the CAMID does not exist, it will be created in the }ClientCAMAssociatedGroups control cube.
CAMIDDefDisplayValue	The alias of the CAMID group.

CellSecurityCubeCreate

CellSecurityCubeCreate creates a security cube from an existing cube using a reduced set of dimensions. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellSecurityCubeCreate (‘DataCube’, ‘0:0:1:0’);
```

Argument	Description
Cube	Name of the data cube.
DimensionMap	String specifying whether the dimension at each position should be used in the security cube. The order of dimensions is the original cube order. A 1 for each included dimension and a 0 for an excluded one. Each value separated by a colon.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the cell security cube.

Example

```
CellSecurityCubeCreate (‘DataCube’, ‘0:0:1:0’);
```

This example creates an RDCLS cube from the cube called Data Cube.

CellSecurityCubeDestroy

CellSecurityCubeDestroy destroys a security cube that was created from an existing cube. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
CellSecurityCubeDestroy (‘DataCube’, ‘0:0:1:0’);
```

Argument	Description
Cube	Name of the data cube.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Destroys the cell security cube.

Example

```
CellSecurityCubeDestroy (‘DataCube’);
```

DeleteClient

DeleteClient deletes a client from the server. Changes applied through the DeleteClient function do not take effect until the Metadata procedure in a process is completed. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DeleteClient(ClientName);
```

Argument	Description
ClientName	The name of the client you want to delete from the server.

Example

```
DeleteClient('Brian');
```

This example removes the client Brian from the server.

DeleteGroup

DeleteGroup deletes a user group from the server. Changes applied through the DeleteGroup function do not take effect until the Metadata procedure in a process is completed. This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
DeleteGroup(GroupName);
```

Argument	Description
GroupName	The group you want to delete.

Example

```
DeleteGroup('Finance');
```

This example deletes the Finance user group from the server.

ElementSecurityGet

ElementSecurityGet retrieves the security level assigned to a specified group for a dimension element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementSecurityGet(DimName, ElName, Group);
```

Argument	Description
DimName	The parent dimension of the element for which you are retrieving a security level.
ElName	The element for which you are retrieving a security level.
Group	The user group for which you are retrieving a security level.

Example

```
ElementSecurityGet('Region', 'Germany', 'Budgeting');
```

This example returns the security level assigned to the Budgeting user group for the Germany element of the Region dimension.

ElementSecurityPut

ElementSecurityPut assigns a security level to a specified group for a dimension element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ElementSecurityPut(Level, DimName, ElName, Group);
```

Argument	Description
Level	The security level you are assigning. There are six possible Level values: • None • Read • Write • Reserve • Lock • Admin
DimName	The parent dimension of the element for which you are assigning a security level.
ElName	The element for which you are assigning a security level.
Group	The user group for which you are assigning a security level.

Example

```
ElementSecurityPut('Reserve', 'Region', 'Germany', 'Budgeting');
```

This example assigns Reserve security to the Budgeting group for the Germany element of the Region dimension.

HierarchyElementSecurityGet

HierarchyElementSecurityGet retrieves the security level assigned to a specified group for a dimension element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementSecurityGet(DimName, HierName, ElName, Group);
```

Argument	Description
DimName	The parent dimension of the element for which you are retrieving a security level.
HierName	The name of the hierarchy within the dimension.
ElName	The element for which you are retrieving a security level.
Group	The user group for which you are retrieving a security level.

Example

```
HierarchyElementSecurityGet('Region', 'Europe', 'Germany', 'Budgeting');
```

This example returns the security level assigned to the Budgeting user group for the Germany element. The element appears in the Europe hierarchy of the Region dimension.

HierarchyElementSecurityPut

HierarchyElementSecurityPut assigns a security level to a specified group for a dimension element.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchyElementSecurityPut(Level, DimName, HierName, ElName, Group);
```

Argument	Description
Level	The security level you are assigning. There are six possible Level values: • None • Read • Write • Reserve • Lock • Admin
DimName	The parent dimension of the element for which you are assigning a security level.
HierName	The name of the hierarchy within the dimension.
ElName	The element for which you are assigning a security level.
Group	The user group for which you are assigning a security level.

Example

```
HierarchyElementSecurityPut('Reserve', 'Region', 'Europe', 'Germany', 'Budgeting');
```

This example assigns Reserve security to the Budgeting group for the Germany element. The element appears in the Europe hierarchy of the Region dimension.

RemoveCAMIDAssociation

RemoveCAMIDAssociation removes all associations between TM1 user groups and a specified CAMID.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RemoveCAMIDAssociation(CAMID, RemoveCAMID);
```

Argument	Description
CAMID	The name of the CAMID group for which you want to remove all security associations.

Argument	Description
RemoveCAMID	Determines if the specified CAMID is deleted from the }ClientCAMAssociatedGroups control cube. 0 leaves the CAMID in the }ClientCAMAssociatedGroups control cube. 1 deletes the CAMID from the }ClientCAMAssociatedGroups control cube.

RemoveCAMIDAssociationFromGroup

RemoveCAMIDAssociationFromGroup removes an association between a TM1 user group and a CAMID.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RemoveCAMIDAssociationFromGroup(GroupName, CAMID);
```

Argument	Description
GroupName	The name of the TM1 user group for which you want to remove the association.
CAMID	The name of the CAMID group for which you want to remove the association.

RemoveClientFromGroup

RemoveClientFromGroup removes a specified client from a user group.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RemoveClientFromGroup(ClientName, GroupName);
```

Argument	Description
ClientName	The client you want to remove.
GroupName	The user group from which you want to remove the client.

Example

```
RemoveClientFromGroup('Brian', 'Finance');
```

This example removes the client Brian from the Finance user group.

SetHierarchyGroupsSecurity

SetHierarchyGroupsSecurity sets the security level for all existing groups for the specified dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetHierarchyGroupsSecurity(securityLevel, dimension, hierarchy)
```

Argument	Description
securityLevel	The security level that you are assigning. There are six possible values: • None • Read • Write • Reserve • Lock • Admin
dimension	Name of the dimension.
hierarchy	Name of the dimension hierarchy.

Example

```
SetHierarchyGroupsSecurity('Reserve', 'Region', 'Europe');
```

This example assigns Reserve security to all existing groups in the Europe hierarchy of the Region dimension.

SetHierarchyElementGroupsSecurity

SetHierarchyElementGroupsSecurity sets the security level for a specified element from a hierarchy in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetHierarchyElementGroupsSecurity(securityLevel, dimension, hierarchy, element)
```

Argument	Description
securityLevel	The security level you are assigning. There are six possible values: • None • Read • Write • Reserve • Lock • Admin
dimension	Name of the dimension.
hierarchy	Name of the dimension hierarchy.

Argument	Description
element	The element for which you are assigning a security level.

Example

```
SetHierarchyElementGroupsSecurity('Reserve', 'Region', 'Europe', 'Germany');
```

This example assigns Reserve security to the Germany element of the Europe hierarchy in the Region dimension.

SetDimensionGroupsSecurity

SetDimensionGroupsSecurity sets the security level for all existing groups for the specified dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetDimensionGroupsSecurity(securityLevel, dimension)
```

Argument	Description
securityLevel	The security level you are assigning. There are six possible values: • None • Read • Write • Reserve • Lock • Admin
dimension	Name of the dimension.

Example

```
SetDimensionGroupsSecurity('Reserve', 'Region');
```

This example assigns Reserve security to all existing groups in the Region dimension.

SetElementGroupsSecurity

SetElementGroupsSecurity sets the security level for a specified element in a dimension.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SetElementGroupsSecurity(securityLevel, dimension, element)
```

Argument	Description
securityLevel	The security level you are assigning. There are six possible values: • None • Read • Write • Reserve • Lock • Admin
dimension	Name of the dimension.
element	The element for which you are assigning a security level.

Example

```
SetElementGroupsSecurity('Reserve', 'Region', 'Germany');
```

This example assigns Reserve security to the Germany element of the Region dimension.

SecurityOverlayGlobalLockCell

SecurityOverlayGlobalLockCell is used to restrict the access rights of a node to read-only by locking it. It uses the global overlay so all users are affected. The overlay cube must be created prior to using this command. The elements provided in the address must be only for the dimensions used in the overlay.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SecurityOverlayGlobalLockCell(bLock, Cube, element1,..., elementN)
```

Argument	Description
bLock	If 1 lock it. 0 unlock it
Cube	Name of the cube.
elementN	Overlay element name that defines the tuple. The order must match the original dimension order of the cube.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the default global security overlay cube. Global overlays apply to all users.

Example

```
SecurityOverlayGlobalLockCell(1,’Sales’,’MA’);
SecurityOverlayGlobalLockCell(0,’Products’,’MA','2011’);
```

In the first example, there is only one dimension used for the overlay. The second example uses two dimensions.

SecurityOverlayCreateGlobalDefault

SecurityOverlayCreateGlobalDefault is used to create or destroy a Security Overlay cube, and to set the overlay for a given area of a data cube.

Creating a data cube with a name that signifies an overlay cube will cause the data cube to be made into an overlay if the server is restarted. When the cube is loaded it will be configured as an overlay if a matching data cube is found.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SecurityOverlayCreateGlobalDefault (Cube,
      DimensionMap)
```

Argument	Description
Cube	Name of the cube.
DimensionMap	String specifying whether the dimension at each position should be used in the overlay. The order of dimensions is the original cube order. A 1 for each included dimension and a 0 for an excluded one. Each value separated by a colon.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the default global security overlay cube. Global overlays apply to all users

Example

```
SecurityOverlayCreateGlobalDefault(‘DataCube’,
        ‘0:0:1:0’);
```

SecurityOverlayDestroyGlobalDefault

SecurityOverlayDestroyGlobalDefault is used to destroy a Security Overlay cube, and to set the overlay for a given area of a data cube.

Creating a data cube with a name that signifies an overlay cube will cause the data cube to be made into an overlay if the server is restarted. When the cube is loaded it will be configured as an overlay if a matching data cube is found

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SecurityOverlayDestroyGlobalDefault (Cube)
```

Argument	Description
Cube	Name of the cube.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the default global security overlay cube. Global overlays apply to all users.

Example

```
SecurityOverlayDestroyGlobalDefault('DataCube');
```

SecurityOverlayGlobalLockNode

SecurityOverlayGlobalLockNode is used to restrict the access rights of a node to read-only by locking it. It uses the global overlay so all users are affected. The overlay cube must be created prior to using this command. The elements provided in the address must be only for the dimensions used in the overlay.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SecurityOverlayGlobalLockNode(bLock, Cube, Address, [AddressDelimiter])
```

Argument	Description
bLock	If 1 lock it. 0 unlock it
Cube	Name of the cube.
Address	Tokenized string sequence of overlay element names that define the tuple. The order must match the original dimension order of the cube.
Address return	Optional character string used to separate element names in the Address parameter. Default value '|'.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the default global security overlay cube. Global overlays apply to all users.

Examples

```
SecurityOverlayGlobalLockNode(1,'Sales','MA');
SecurityOverlayGlobalLockNode(0,'Products','MA | 2011');
SecurityOverlayGlobalLockNode(0,'Products', 'MA : 2011', ':');
```

In the first example there is only one dimension used for the overlay. The other two examples use two dimensions.

SecurityRefresh

SecurityRefresh reads all the security control cubes and regenerates the internal structures in the server that are used by TM1 API functions.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SecurityRefresh;
```

Arguments

None.

Server Manipulation TurboIntegrator Functions

These functions facilitate server manipulation.

BatchUpdateFinish

BatchUpdateFinish instructs the server to exit batch update mode.

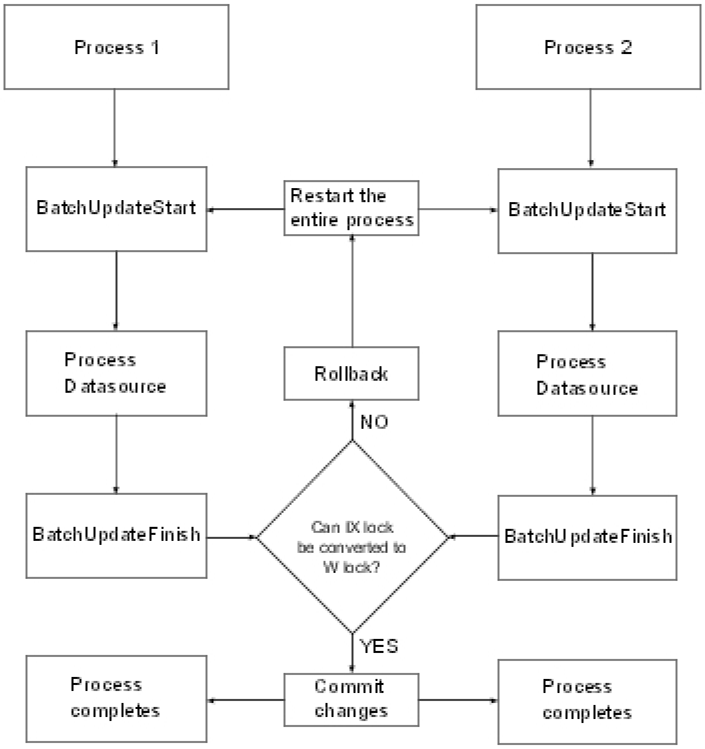
This function is valid in TM1 TurboIntegrator processes only.

Semantics

When multiple processes are running in batch update mode and applying changes to a single cube, the TM1 locking scheme may prevent one of the processes from updating the cube. This is by design; when one process obtains a lock to write changes to a cube, other processes will be prevented from writing to that cube in the interest of maintaining data integrity.

This locking scheme can be illustrated using an example of two processes, Process 1 and Process 2, that update a single cube.

- Both processes start and call the BatchUpdateStart function to initiate batch updates.
- Each process operates on a unique data source.
- Process 1 completes processing data and calls the BatchUpdateFinish function. The process obtains a write lock to the cube and commits changes.
- While Process 1 still holds a write lock to the cube, Process 2 completes processing data and calls the BatchUpdateFinish function. However, because Process 1 retains the lock, Process 2 cannot obtain a lock to the cube. All data changes applied in Process 2 are rolled back and Process 2 is restarted. This ensures data integrity.



Depending on the size of the datasource for Process 2, the data rollback and process re-execution can cause a noticeable decrease in performance. To address this performance issue, consider using the BatchUpdateFinishWait function in place of BatchUpdateFinish.

Syntax

```
BatchUpdateFinish(SaveChanges);
```

Argument	Description
SaveChanges	A flag that instructs the server to either save or discard changes committed while in batch update mode. Specify 0 to save changes, 1 to discard changes.

Example

```
BatchUpdateFinish(0);
```

This example instructs the server to save changes to TM1 data and exit batch update mode.

BatchUpdateFinishWait

BatchUpdateFinishWait is identical to BatchUpdateFinish except the process waits until the lock becomes available and then commits changes. If a process calls BatchUpdateFinishWait but is unable to secure a cube write lock to commit changes, the process waits until the lock becomes available and then commits changes.

This function is valid in TM1 TurboIntegrator processes only.

Data changes applied in the process are not rolled back and the process is not re-executed.

Note: While waiting for the cube write lock, the process releases any read locks it acquired for other objects during process execution. Because these read locks are released before the process can commit changes to the cube, the objects for which the read locks are released can be modified *before* the cube is updated. This can lead to data inconsistency when using BatchUpdateFinishWait.

We recommend that BatchUpdateFinishWait be used only in controlled situations where you know that other processes are not modifying data or metadata related to the process that calls BatchUpdateFinishWait.

Syntax

```
BatchUpdateFinishWait(SaveChanges);
```

Argument	Description
SaveChanges	A flag that instructs the server to either save or discard changes committed while in batch update mode. Specify 0 to save changes, 1 to discard changes.

Example

```
BatchUpdateFinishWait(0);
```

This example instructs the server to save changes to TM1 data and exit batch update mode.

BatchUpdateStart

BatchUpdateStart enables batch updates.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
BatchUpdateStart;
```

Arguments

None.

DisableBulkLoadMode

DisableBulkLoadMode disables bulk load processing.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

See “EnableBulkLoadMode” on page 574 for details.

EnableBulkLoadMode

EnableBulkLoadMode enables bulk load processing for a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

You can enable Bulk Load Mode in either the Prolog or Epilog section of a TurboIntegrator process. For efficiency, enable Bulk Load Mode in the first, or very close to the first, statement in the Prolog section of your process.

After enabling Bulk Load Mode in a process, it can only be disabled on the last line in the Epilog section. If you attempt to disable Bulk Load Mode anywhere else in the process, the process will not compile.

If the mode is enabled in one TurboIntegrator process, it remains enabled until explicitly disabled or until the chore completes. This means you can enable the mode in a process within a chore and then run a series of TurboIntegrator processes before disabling it. You can also enter and exit Bulk Load Mode repeatedly, using the mode only for certain critical parts of a chore.

Use the following TurboIntegrator commands to enable and disable Bulk Load Mode in a TurboIntegrator process.

```
EnableBulkLoadMode();
```

Use the following TurboIntegrator function only on the last line in the Epilog section of your TI process when using Bulk Load Mode.

```
DisableBulkLoadMode();
```

RefreshMdxHierarchy

RefreshMdxHierarchy updates the MDX hierarchies in a server without requiring you to restart the server.

Use this function after configuring or editing the custom named hierarchy levels for a dimension in the }HierarchyProperties control cube.

For details on using named levels with dimensions, see the related section in the IBM Cognos *TM1 for Developers* documentation.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
RefreshMdxHierarchy(dimensionName, hierarchy)
```

Argument	Description
dimensionName	Optional string parameter to specify a specific dimension to update. Leave this parameter blank to update all dimensions.
hierarchy	The name of the hierarchy within the dimension. This is an optional parameter.

Example

Update all dimensions:

```
RefreshMdxHierarchy('');
```

To update only the customers dimension:

```
RefreshMdxHierarchy('customers');
```

SaveDataAll

SaveDataAll saves all TM1 data from server memory to disk and restarts the log file.

This function is valid in TM1 TurboIntegrator processes only.

Using SaveDataAll in a Chore

SaveDataAll commits all changes a chore makes prior to calling the SaveDataAll function.

While a chore is running, it accumulates locks on the objects it accesses. The commit operation initiated by the SaveDataAll function temporarily releases all these locks. Once the commit is complete, SaveDataAll reacquires all the locks it had before so it can continue to access the objects it was working on.

There is a brief window during the commit operation where the locks are released and another user or TurboIntegrator process could delete objects the original chore was using. When the original chore attempts to reacquire the locks on those objects, the objects will not be available and the chore will cease processing. In this case, an error similar to the following is written to the Tm1s.log file:

```
844 WARN 2008-04-01 16:40:09,734 TM1.Server TM1ServerImpl::FileSave could
not reacquire lock on object with index 0x200002ca
```

Lock contention and using SaveDataAll at the end of TurboIntegrator processes

Using SaveDataAll as last command in a TurboIntegrator process can increase lock contention in TM1 TurboIntegrator processes.

In IBM TM1 versions, SaveDataAll was often added to the end of a TurboIntegrator process that loads data with logging disabled. The SaveDataAll provided a way to write data from memory to disk directly after a successful import, so that the newly imported data would not be lost in case of a mishap, such as a server crash.

However, adding SaveDataAll as the last command can result in numerous TurboIntegrator import processes, each one with SaveDataAll as last command. This technique worked in TM1 Version 9.0 and older due to the previous lock model which used only the global write lock. At any given time in earlier versions only one write operation could take place. Therefore competing concurrent SaveDataAll operations never occurred from multiple concurrent write operations.

Version 9.1 and newer introduced a more granular lock-by-object model that enables concurrent write operations, if these write operations do not compete for the same resources. If they do compete for the same resources, a lock contention occurs forcing one of the processes to rollback. So now two TurboIntegrator import processes may run simultaneously if they do not share any objects, for example, if they import into two different cubes.

The TurboIntegrator function SaveDataAll relies on the transaction logfile tm1s.log and involves all objects within a data model. Therefore, two TurboIntegrator import processes, both using the function SaveDataAll, cannot run in parallel: one will be executed, the other one (and its TurboIntegrator process) will be forced to rollback. The same is true if the TurboIntegrator processes are part of chores: only one chore will proceed to execute the TurboIntegrator function SaveDataAll, the other chore will be forced to rollback.

A rollback is undesirable from a performance point of view, as it increases the total execution time of a TurboIntegrator process or chore. Competing concurrent SaveDataAll operations will always lead to a lock contention and to a rollback.

There are two possible solutions to avoid competing concurrent SaveDataAll operations:

- Do not use the TurboIntegrator function SaveDataAll. Instead enable Cube Logging for the import cubes.
- If enabling Cube Logging for the import cubes cannot be done for performance reasons, within the TM1 application there should be only one process calling the TurboIntegrator function SaveDataAll. Use a stand-alone, single, distinct chore to execute the SaveDataAll operation.

Syntax

```
SaveDataAll;
```

Arguments

None.

ServerShutdown

ServerShutdown shuts down a server running as an application. ServerShutdown cannot be used to shut down a server running as a Windows service.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ServerShutDown(SaveData);
```

Argument	Description
SaveData	A Boolean value that indicates whether the server should save changes to disk before shutting down. If SaveData = 0, the server shuts down without saving changes. If SaveData = 1, the server saves changes from memory to disk before shutting down.

Example

```
ServerShutdown(1);
```

This example shuts down the server and saves data to disk.

Subset Manipulation TurboIntegrator Functions

These functions facilitate subset manipulation.

HierarchySubsetAliasSet

HierarchySubsetAliasSet sets the alias attribute to be used in a subset. HierarchySubsetAliasSet returns 1 if successful, 0 otherwise.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetAliasSet(DimName, HierName, SubName, AliasName);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to set the alias.
HierName	The name of the hierarchy within the dimension.
SubName	The subset for which you want to set the alias.
AliasName	The alias you want to use in the subset.

HierarchySubsetCreate

HierarchySubsetCreate creates an empty public subset of a specified hierarchy and dimension.

When the optional `AsTemporary` argument is set to 1, the subset is temporary and persists only for the duration of the TurboIntegrator process or chore in which the subset is created.

Note:

For TM1 Server version 11.2.0 and earlier, temporary subsets were visible and usable only by the process that created it and any of its child processes. Temporary subsets were not visible to the ancestor and sibling processes. You could create same-named subsets in sibling child processes with the same parent process.

For TM1 Server version 11.3.0 and later, these temporary subsets are visible to the ancestor and sibling processes. If a parent TurboIntegrator process A invokes two child TurboIntegrator processes A1 and A2, and the child TurboIntegrator process A1 creates a temporary subset S, the temporary subset S exists for the duration of the parent TurboIntegrator process A. You cannot create a temporary subset with the same name S in the sibling TurboIntegrator process A2 since the subset is visible and usable by siblings A1 and A2.

While a temporary subset exists, the temporary subset takes precedence over any same-named public subset. If another TurboIntegrator function references a subset that exists in both a temporary and permanent state, the function operates upon the temporary subset.

There is no locking associated with a temporary subset, as a temporary subset is never saved. This can result in improved performance, because there is no need for TurboIntegrator to wait for locks to be released before operating upon a temporary subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetCreate(DimName, HierName, SubName, [AsTemporary]);
```

Argument	Description
DimName	The parent dimension of the subset you are creating.
HierName	The name of the hierarchy within the dimension.
SubName	The name you want to assign to the subset.
AsTemporary	This is an optional argument that specifies whether the subset being created is temporary. 1 indicates a temporary subset, 0 indicates a permanent subset. If this argument is omitted, the subset is permanent.

Example

```
HierarchySubsetCreate('Region', 'European', 'Northern Europe', 1);
```

This example creates the temporary Northern Europe subset of the European hierarchy in the Region dimension. You can use SubsetElementInsert to add elements to the subset.

HierarchySubsetDeleteAllElements

HierarchySubsetDeleteAllElements deletes all elements from a public subset of a dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetDeleteAllElements(DimName, HierName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete elements.

Argument	Description
HierName	The name of the hierarchy within the dimension.
SubsetName	The subset from which you want to delete elements. This must be a public subset. TurboIntegrator cannot access private objects.

Example

```
HierarchySubsetDeleteAllElements('Region', 'European', 'Central Europe');
```

This example deletes all elements from the Central Europe subset of the European hierarchy in the Region dimension.

HierarchySubsetDestroy

HierarchySubsetDestroy deletes a subset from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetDestroy(DimName, HierName, SubName);
```

Argument	Description
DimName	The parent dimension of the subset you are deleting.
HierName	The name of the hierarchy within the dimension.
SubName	The name of the subset you want to delete.

Example

```
HierarchySubsetDestroy('Region', 'European', 'Northern Europe');
```

This example deletes the Northern Europe subset of the European hierarchy in the Region dimension.

HierarchySubsetElementExists

HierarchySubsetElementExists determines whether a specific element exists within a specific public subset on the server from which a TurboIntegrator process is executed. HierarchySubsetElementExists cannot be used to determine if an element exists in a private subset.

If the element exists in the specified subset, the function returns 1, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetElementExists(DimName, HierName, SubsetName, ElementName);
```

Argument	Description
DimName	The dimension parent of the subset containing the element whose existence you want to confirm.
HierName	The name of the hierarchy in the specified dimension.

Argument	Description
SubsetName	The public subset containing the element whose existence you want to confirm.
ElementName	The element whose existence you want to confirm.

Example

```
HierarchySubsetElementExists('Region', 'Eastern', 'Europe', 'Italy');
```

This example determines if the Italy element exists in the Europe subset of the Eastern hierarchy from the Region dimension.

HierarchySubsetElementDelete

HierarchySubsetElementDelete deletes an element from a subset of a dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetElementDelete(DimName, HierName, SubName, Index);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete an element.
HierName	The name of the hierarchy within the dimension.
SubName	The subset from which you want to delete an element.
Index	The index number of the element you want to delete from the subset.

Example

```
HierarchySubsetElementDelete('Region', 'European', 'Northern Europe', 3);
```

This example deletes the third element from the Northern Europe subset of the European hierarchy in the Region dimension.

HierarchySubsetElementGetIndex

HierarchySubsetElementGetIndex retrieves the index of an element in a subset of a dimension hierarchy.

The function returns the index of the first occurrence of the specified element. If the element does not exist in the subset or cannot be found, then zero is returned. If the dimension or subset cannot be found or an out-of-range start index is specified, then an error is thrown and the TurboIntegrator function is stopped.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetElementGetIndex(DimName, HierName, SubsetName, ElementName, StartIndex);
```

Argument	Description
DimName	The parent dimension of the subset.

Argument	Description
HierName	The name of the hierarchy within the dimension.
SubsetName	The subset that contains the element.
ElementName	The element name to search for in the subset.
StartIndex	The index number to begin searching from. The value must be between 1 and the size of the subset.

Example

```
HierarchySubsetElementGetIndex('Region', 'Country', 'Europe', 'Italy', 3);
```

This example retrieves the index for Italy from the Europe subset of the Country hierarchy in the Region dimension. The search starts at index 3.

HierarchySubsetElementInsert

HierarchySubsetElementInsert adds an element to an existing subset in a dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetElementInsert(DimName, HierName, SubName, ElName, Position);
```

Argument	Description
DimName	The parent dimension of the subset to which you want to add an element.
HierName	The name of the hierarchy within the dimension.
SubName	The name of the subset to which you are adding an element.
ElName	The name of the element you want to add to the subset. The element must exist in the TM1 database.
Position	A value that indicates the index position of the element within the subset.

Example

```
HierarchySubsetElementInsert('Region', 'European', 'Northern Europe', 'Finland',3);
```

This example adds the element Finland to the Northern Europe subset of the European hierarchy in the Region dimension. Finland is the third element in the subset definition.

HierarchySubsetExists

HierarchySubsetExists determines if a specific public subset exists on the server from which a TurboIntegrator process is executed. The function returns 1 if the subset exists on the server, otherwise it returns 0. Note that this function cannot be used to determine the existence of private subsets.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetExists(DimName, HierName, SubsetName);
```

Argument	Description
DimName	The name of the dimension that is the parent of the subset whose existence you want to confirm.
HierName	The name of the hierarchy within the dimension.
SubsetName	The name of the public subset whose existence you want to confirm

Example

```
HierarchySubsetExists('Region', 'Industrialized, 'Northern Europe');
```

This example determines if the Northern Europe subset exists within the Industrialized hierarchy of the Region dimension.

HierarchySubsetGetSize

HierarchySubsetGetSize returns the number of elements in a subset of a dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetGetSize(DimName, HierName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to determine size.
HierName	The name of the hierarchy within the dimension.
SubsetName	The subset for which you want to determine size.

Example

```
HierarchySubsetGetSize('Region', 'Eastern', 'EurAsia');
```

This function returns the number of elements in the EurAsia subset of the Eastern hierarchy in the Region dimension.

HierarchySubsetGetElementName

HierarchySubsetGetElementName returns the name of the element at a specified index location within a given subset of a dimension hierarchy.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetGetElementName(DimName, HierName, SubsetName, ElementIndex);
```

Argument	Description
DimName	The parent of the subset from which you want to retrieve an element name.

Argument	Description
HierName	The name of the hierarchy within the dimension.
SubsetName	The subset from which you want to retrieve an element name.
ElementIndex	A number representing the position within the subset of the element you want to retrieve.

Example

```
HierarchySubsetGetElementName('Region', 'Western', 'Americas', 4);
```

This example returns the name of the fourth element in the Americas subset of the Western hierarchy in Region dimension.

HierarchySubsetIsAllSet

HierarchySubsetIsAllSet sets a subset to use all elements of the parent dimension. HierarchySubsetIsAllSet returns 1 if successful, 0 otherwise.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetIsAllSet(DimName, HierName, SubName, Flag);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to use all elements.
HierName	The name of the hierarchy within the dimension.
SubName	The subset for which you want to use all dimension elements.
Flag	Any non-zero value specifies that the subset uses all the current elements from the parent dimension and will dynamically update to use all elements from the parent dimension whenever the subset is called. Specifying a zero value freezes the elements in the subset as the current set of all elements in the parent dimension. The subset will not dynamically update to use all dimension elements in the future.

HierarchySubsetMDXGet

HierarchySubsetMDXGet retrieves the MDX expression used to create a subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetMDXGet(DimName, HierName, SubName);
```

Argument	Description
DimName	The parent dimension of the subset.

Argument	Description
HierName	The name of the hierarchy within the dimension.
SubName	The subset for which you want to retrieve the MDX expression.

Example

```
mdxString = HierarchySubsetMDXGet('Cities', 'Italy', 'testsubset');
```

HierarchySubsetMDXSet

HierarchySubsetMDXSet applies a specified MDX expression to an existing public subset of a hierarchy.

If the passed MDX expression is valid, the specified subset is saved as a dynamic subset defined by the MDX expression.

If the passed MDX expression is an empty string, the subset is converted to a static subset that contains the elements that are in place when HierarchySubsetMDXSet is executed.

The function returns the number of elements that the subset contains.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
HierarchySubsetMDXSet(DimName, HierName, SubName, MDX_expression);
```

Argument	Description
DimName	The parent dimension of the subset.
HierName	The name of the hierarchy within the dimension.
SubName	The subset to which you want to apply the MDX expression. SubName must be a public subset. If this subset does not exist, an error is logged.
MDX_expression	The MDX expression that you want to apply to the subset. If the MDX expression is invalid, TurboIntegrator processing stops, the subset is not modified, and an error is logged. If the MDX_expression argument is an empty string, the subset is converted to a static subset.

Example

```
HierarchySubsetMDXSet('Cities', 'World', 'Sub1', '{ [Cities].[Cities].[level000].members }');
```

This example updates the `Sub1` subset of the `World` hierarchy to a dynamic subset that contains the current leaf elements of the `Cities` dimension. When leaf elements are added or removed from the `Cities` dimension, the `mySub1` subset is dynamically updated to reflect the changes in the parent dimension.

SubsetAliasSet

SubsetAliasSet sets the alias attribute to be used in a subset. SubsetAliasSet returns 1 if successful, 0 otherwise.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetAliasSet( DimName, SubName, AliasName );
```

Argument	Description
DimName	The parent dimension of the subset for which you want to set the alias.
SubName	The subset for which you want to set the alias.
AliasName	The alias you want to use in the subset.

SubsetCreate

SubsetCreate creates an empty public subset of a specified dimension.

When the `AsTemporary` argument is set to 1, the subset is temporary and persists only for the duration of the TurboIntegrator process or a single-commit chore in which the subset is created. If a parent TurboIntegrator process invokes child TurboIntegrator processes by using the `ExecuteProcess` or `ExecuteProcessWithReturn` function, and the temporary subset is created in one of these child TurboIntegrator processes, the subset persists for the duration of the parent TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Note:

For TM1 Server version 11.2.0 and earlier, temporary views and subsets were visible and usable only by the process that created it and any of its child processes. Temporary views and subsets were not visible to the ancestor and sibling processes. You could create same-named subsets in sibling child processes with the same parent process.

For TM1 Server version 11.3.0 and later, these temporary subsets are visible to the ancestor and sibling processes. If a parent TurboIntegrator process A invokes two child TurboIntegrator processes A1 and A2, and the child TurboIntegrator process A1 creates a temporary subset S, the temporary subset S exists for the duration of the parent TurboIntegrator process A. You cannot create a temporary subset with the same name S in the sibling TurboIntegrator process A2 since the subset is visible and usable by siblings A1 and A2.

A chore is a special case of a parent TurboIntegrator process that invokes a child TurboIntegrator process that is scheduled to run at a specific time. You can use two types of chores.

Single-commit
: Within the scope / execution tree of a single-commit chore, a temporary subset of the same name can be created only for one child TurboIntegrator process.

Multi-commit
: Within the scope / execution tree of a multi-commit chore, which commits after every child TurboIntegrator process, every child TurboIntegrator process can create a temporary subset of the same name because a temporary subset will not persist after a commit.

While a temporary subset exists, the temporary subset takes precedence over any same-named public or private subset. If another TurboIntegrator function references a subset that exists in both a temporary and permanent state, the function operates upon the temporary subset.

There is no locking associated with a temporary subset because a temporary subset is never saved. This can result in improved performance because there is no need for TurboIntegrator to wait for locks to be released before operating upon a temporary subset.

Syntax

```
SubsetCreate(DimName, SubName, [AsTemporary]);
```

Argument	Description
DimName	The parent dimension of the subset you are creating.
SubName	The name that you want to assign to the subset.
AsTemporary	This is an optional argument that specifies whether the subset that is being created is temporary. 1 indicates a temporary subset. 0 indicates a permanent subset. If this argument is omitted, the subset is permanent.

Example

```
SubsetCreate('Region', 'Northern Europe', 1);
```

This example creates the temporary Northern Europe subset of the Region dimension. You can use `SubsetElementInsert` to add elements to the subset.

SubsetCreateByMDX

SubsetCreateByMDX creates a public subset based on a passed MDX expression.

When the `AsTemporary` argument is set to 1, the subset is temporary and persists for the duration of the TurboIntegrator process or a single-commit chore in which the subset is created. If a parent TurboIntegrator process invokes child TurboIntegrator processes by using the `ExecuteProcess` or `ExecuteProcessWithReturn` function, and the temporary subset is created in one of these child TurboIntegrator processes, the subset persists for the duration of the parent TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Note:

For TM1 Server version 11.2.0 and earlier, temporary views and subsets were visible and usable only by the process that created it and any of its child processes. Temporary views and subsets were not visible to the ancestor and sibling processes. You could create same-named subsets in sibling child processes with the same parent process.

For TM1 Server version 11.3.0 and later, these temporary subsets are visible to the ancestor and sibling processes. If a parent TurboIntegrator process A invokes two child TurboIntegrator processes A1 and A2, and the child TurboIntegrator process A1 creates a temporary subset S, the temporary subset S exists for the duration of the parent TurboIntegrator process A. You cannot create a temporary subset with the same name S in the sibling TurboIntegrator process A2 since the subset is visible and usable by siblings A1 and A2.

A chore is a special case of a parent TurboIntegrator process that invokes a child TurboIntegrator process. You can use two types of chores.

Single-commit
: Within the scope / execution tree of a single-commit chore, a temporary subset of the same name can be created only for one child TurboIntegrator process.

Multi-commit
: Within the scope / execution tree of a multi-commit chore, which commits after every child TurboIntegrator process, every child TurboIntegrator process can create a temporary subset of the same name because a temporary subset will not persist after a commit.

While a temporary subset exists, the temporary subset takes precedence over any same-named public or private subset. If another TurboIntegrator function references a subset that exists in both a temporary and permanent state, the function operates upon the temporary subset.

There is no locking associated with a temporary subset because a temporary subset is never saved. This can result in improved performance because there is no need for TurboIntegrator to wait for locks to be released before operating upon a temporary subset.

Syntax

```
SubsetCreatebyMDX(SubName, MDX_Expression, [AsTemporary]);
```

Argument	Description
SubName	The name you want to assign to the subset.
MDX_Expression	An MDX expression that returns a subset.
AsTemporary	This is an optional argument that specifies whether the subset that is being created is temporary. 1 indicates a temporary subset. 0 indicates a permanent subset. If this argument is omitted, the subset is permanent.

Example

```
SubsetCreatebyMDX('0-level months', '{TM1SORT( {TM1FILTERBYLEVEL(
{TM1SUBSETALL([month] )}, 0)}, ASC)} ', 1 );
```

This example creates a temporary subset that is named '0-level months' based on an MDX expression that returns a subset that consists of all 0-level elements in the Month dimension, which is sorted in ascending alphabetical order.

Example of temporary subset used by parent process

In this example, two processes are created. The parent process, Process-A, calls the child process, Process-B. Process-B creates a temporary subset that Process-A uses as the data source.

Process-A/prolog:

```
ExecuteProcess('Process-B');
DatasourceDimensionSubset = 'My2003Months';
```

Process-B/prolog:

```
SubsetCreateByMdx('My2003Months', '{[plan_time].[Jan-2003],[plan_time].[Jul-2003]}', 1);
```

SubsetDeleteAllElements

SubsetDeleteAllElements deletes all elements from a public subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetDeleteAllElements(DimName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete elements.

Argument	Description
SubsetName	The subset from which you want to delete elements. This must be a public subset. TurboIntegrator cannot access private objects.

Example

```
SubsetDeleteAllElements('Region', 'Central Europe');
```

This example deletes all elements from the Central Europe subset of the Region dimension.

SubsetDestroy

SubsetDestroy deletes a subset from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetDestroy(DimName, SubName);
```

Argument	Description
DimName	The parent dimension of the subset you are deleting.
SubName	The name of the subset you want to delete.

Example

```
SubsetDestroy('Region', 'Northern Europe');
```

This example deletes the Northern Europe subset of the Region dimension.

SubsetElementDelete

SubsetElementDelete deletes an element from a subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetElementDelete(DimName, SubName, Index);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete an element.
SubName	The subset from which you want to delete an element.
Index	The index number of the element you want to delete from the subset.

Example

```
SubsetElementDelete('Region', 'Northern Europe', 3);
```

This example deletes the third element from the Northern Europe subset of the Region dimension.

SubsetElementExists

SubsetElementExists determines whether a specific element exists within a specific public subset on the server from which a TurboIntegrator process is executed. SubsetElementExists cannot be used to determine if an element exists in a private subset.

If the element exists in the specified subset, the function returns 1, otherwise it returns 0.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetElementExists(DimName, SubsetName, ElementName);
```

Argument	Description
DimName	The dimension parent of the subset containing the element whose existence you want to confirm.
SubsetName	The public subset containing the element whose existence you want to confirm.
ElementName	The element whose existence you want to confirm.

Example

```
SubsetElementExists('Region', 'Europe', 'Italy');
```

This example determines if the Italy element exists in the Europe subset of the Region dimension.

SubsetElementGetIndex

SubsetElementGetIndex retrieves the index of an element in a subset. The function returns the index of the first occurrence of the specified element.

If the element does not exist in the subset or cannot be found, then zero is returned. If the dimension or subset cannot be found or an out-of-range start index is specified, then an error is thrown and the TurboIntegrator function is stopped.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetElementGetIndex(DimName, SubsetName, ElementName, StartIndex);
```

Argument	Description
DimName	The parent dimension of the subset.
SubsetName	The subset that contains the element.
ElementName	The element name (or alias) to search for in the subset.
StartIndex	The index number to begin searching from. The value must be between 1 and the size of the subset.

Example

```
SubsetElementGetIndex('Region', 'Europe', 'Italy', 3);
```

This example retrieves the index for Italy from the Europe subset of the Region dimension. The search starts at index 3.

SubsetElementInsert

SubsetElementInsert adds an element to an existing subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetElementInsert(DimName, SubName, ElName, Position);
```

Argument	Description
DimName	The parent dimension of the subset to which you want to add an element.
SubName	The name of the subset to which you are adding an element.
ElName	The name of the element you want to add to the subset. The element must exist in the TM1 database.
Position	A value that indicates the index position of the element within the subset.

Example

```
SubsetElementInsert('Region', 'Northern Europe', 'Finland',3);
```

This example adds the element Finland to the Northern Europe subset of the Region dimension. Finland is the third element in the subset definition.

SubsetExists

SubsetExists determines whether a specific public subset exists on the server from which a TurboIntegrator process is executed.

The function returns 1 if the subset exists on the server, otherwise it returns 0. Note that this function cannot be used to determine the existence of private subsets.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetExists(DimName, SubsetName);
```

Argument	Description
DimName	The name of the dimension that is the parent of the subset whose existence you want to confirm.
SubsetName	The name of the public subset whose existence you want to confirm

Example

```
SubsetExists('Region', 'Northern Europe');
```

This example determines if Northern Europe subset of the Region dimension exists on the server.

SubsetExpandAboveSet

SubsetExpandAboveSet sets the Expand Above property for a subset. The function returns 1 if successful, otherwise it returns 0.

When this property is set to TRUE, children of a consolidation are displayed above the consolidation when the consolidation displays on a row, and to the left of the consolidation when the consolidation displays on a column.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetExpandAboveSet( DimName, SubsetName, ExpandAboveFlag);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to set the Expand Above property.
SubsetName	The subset for which you want to set the Expand Above property.
ExpandAboveFlag	Set ExpandAboveFlag to 1 to set the Expand Above property to TRUE. When this property is TRUE, consolidations expand above on rows and to the left on columns. Set ExpandAboveFlag to 0 to set the Expand Above property to FALSE. When this property is FALSE, consolidations expand below on rows and to the right on columns.

Example

```
SubsetExpandAboveSet('Region', 'Europe', 1 );
```

This example sets the Expand Above property to TRUE for the Europe subset of the Region dimension.

SubsetFormatStyleSet

SubsetFormatStyleSet applies an existing display style to a named subset.

Display styles are defined for specific elements. If you apply an existing display style to a subset that includes elements that are not included in the display style, no formatting is applied to those elements.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetFormatStyleSet( DimName, SubsetName, FormatName);
```

Argument	Description
DimName	The parent dimension of the subset to which you want to apply a display style.
SubsetName	The name of the subset to which you are applying a display style.
FormatName	The name of the existing display style you want to apply to the subset.

Example

```
SubsetFormatStyleSet ('Region', 'Northern Europe', 'BoldCurrencyLeftJustified');
```

This example applies the BoldCurrencyLeftJustified display style to the Northern Europe subset of the Region dimension.

SubsetGetElementName

SubsetGetElementName returns the name of the element at a specified index location within a given subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetGetElementName(DimName, SubsetName, ElementIndex);
```

Argument	Description
DimName	The parent of the subset from which you want to retrieve an element name.
SubsetName	The subset from which you want to retrieve an element name.
ElementIndex	A number representing the position within the subset of the element you want to retrieve.

Example

```
SubsetGetElementName('Region', 'Americas', 4);
```

This example returns the name of the fourth element in the Americas subset of the Region dimension.

SubsetGetSize

SubsetGetSize returns the number of elements in a subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetGetSize(DimName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to determine size.
SubsetName	The subset for which you want to determine size.

Example

```
SubsetGetSize('Region', 'EurAsia');
```

This function returns the number of elements in the EurAsia subset of the Region dimension.

SubsetIsAllSet

SubsetIsAllSet sets a subset to use all elements of the parent dimension. SubsetIsAllSet returns 1 if successful, 0 otherwise.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetIsAllSet(DimName, SubName, Flag);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to use all elements.
SubName	The subset for which you want to use all dimension elements.
Flag	Any non-zero value specifies that the subset uses all the current elements from the parent dimension and will dynamically update to use all elements from the parent dimension whenever the subset is called. Specifying a zero value freezes the elements in the subset as the current set of all elements in the parent dimension. The subset will not dynamically update to use all dimension elements in the future.

SubsetMDXGet

SubsetMDXGet retrieves the MDX expression used to create a subset.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetMDXGet(DimName, SubName);
```

Argument	Description
DimName	The parent dimension of the subset.
SubName	The subset for which you want to retrieve the MDX expression.

Example

```
mdxString = SubsetMDXGet('Cities', 'testsubset');
```

SubsetMDXSet

SubsetMDXSet applies a specified MDX expression to an existing public subset.

If the passed MDX expression is valid, the specified subset is saved as a dynamic subset defined by the MDX expression.

If the passed MDX expression is an empty string, the subset is converted to a static subset that contains the elements that are in place when SubsetMDXSet is executed.

The function returns the number of elements that the subset contains.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
SubsetMDXSet(DimName, SubName, MDX_expression);
```

Argument	Description
DimName	The parent dimension of the subset.
SubName	The subset to which you want to apply the MDX expression. SubName must be a public subset. If this subset does not exist, an error is logged.
MDX_expression	The MDX expression that you want to apply to the subset. If the MDX expression is invalid, TurboIntegrator processing stops, the subset is not modified, and an error is logged. If the MDX_expression argument is an empty string, the subset is converted to a static subset.

Examples

```
SubsetMdxSet( 'YZProducts', 'mySub1', '{ Filter( [YZProducts].[YZProducts].members, 
IsLeaf( [YZProducts].[YZProducts].currentmember ) ) }' );
```

This example updates the `mySub1` subset to a dynamic subset that contains the current leaf elements of the `YZProducts` dimension. When leaf elements are added or removed from the `YZProduct` dimension, the `mySub1` subset is dynamically updated to reflect the changes in the parent dimension.

One possible use of the SubsetMDXSet function is to apply an MDX expression to update an existing subset, and then immediately convert the subset to static.

```
SubsetMDXSet( 'YZProducts', 'mySub1', '{ [YZProducts].[YZProducts].[level000].members }' );
SubsetMDXSet( 'YZProducts', 'mySub1', '' );
```

This two-call sequence updates the `mySub1` subset to a static subset that contains the current top-level elements of the `YZProducts` dimension.

The first call of SubsetMDXSet applies the `{ [YZProducts].[YZProducts].[level000].members }` MDX expression to the `mysub1` subset, resulting in a dynamic subset that includes all top-level (level 0) elements of the `YZProducts` dimension.

The second call of SubsetMDXSet passes an empty string as the MDX_expression argument, so the `mysub1` subset is converted to a static subset.

View Manipulation TurboIntegrator Functions

These functions pertain to view manipulation.

PublishView

PublishView publishes a named private view on the server.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
PublishView(Cube, View, PublishPrivateSubsets, OverwriteExistingView);
```

Argument	Description
Cube	The name of the cube containing the private view to be published.
View	The name of the private view to be published.

Argument	Description
PublishPrivateSubsets	This Boolean argument (1 or 0) determines if any private subsets present in the view should also be published. If PublishPrivateSubsets is true (1) , all private subsets used in the view are published along with the view. If this argument is false (0) , private subsets are not published. A public view cannot contain private subsets, so the view will not be published and an error will be written to the TurboIntegrator log file. **Note:** If a private subset contains another private subset as a user-defined consolidation, the subset can never be published using the PublishView function, regardless of the value of the PublishPrivateSubsets argument.
OverwriteExistingView	This Boolean argument (1 or 0) determines if any existing identically named public view should be overwritten when the private view is published. If OverwriteExistingView is true (1) , any existing identically named public view will be overwritten when the private view is published. If this argument is false (0), the public view will not be overwritten, the private view will not be published, and an error will be written to the TurboIntegrator log file.

DisableMTQViewConstruct

DisableMTQViewConstruct disables multi-threaded query processing when calculating a view to be used as a TurboIntegrator datasource for a single TurboIntegrator process. When MTQQuery=T in the tms1.cfg file, DisableMTQViewConstruct can be called to override this value on a TurboIntegrator process.

This function must appear in the Prolog, it has no effect in any other procedure within a process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

Note: If the value of the **`MTQ`** parameter is 1 (or OFF), this functionality is turned off entirely and cannot be overridden.

The value of **`MTQQuery`** can be overridden on a single TurboIntegrator process by calling the DisableMTQViewConstruct function.

If **`MTQQuery`**=T (the default), DisableMTQViewConstruct can be called to disable the functionality for individual TurboIntegrator processes.

After enabling EnableMTQViewConstruct in a process, it can only be disabled on the last line in the Epilog section. If you attempt to use DisableMTQViewConstruct anywhere else in the process, the process will not compile.

If the mode is enabled in one TurboIntegrator process, it remains enabled until explicitly disabled or until the chore completes. This means you can enable the mode in a process and then run a series of TurboIntegrator processes before disabling it.

Example

Use the following TurboIntegrator commands to disable multi-threaded query processing when calculating a view to be used as a TurboIntegrator datasource for a single TurboIntegrator process.

```
DisableMTQViewConstruct()
```

See also “EnableMTQViewConstruct” on page 596.

EnableMTQViewConstruct

EnableMTQViewConstruct enables multi-threaded query processing when calculating a view to be used as a TurboIntegrator datasource for a single TurboIntegrator process. When MTQQuery=F in the tms1.cfg file, EnableMTQViewConstruct can be called to override this value on a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

Note: If the value of the **MTQ** parameter is 1 (or OFF), this functionality is turned off entirely and cannot be overridden.

The value of **MTQQuery** can be overridden on a single TurboIntegrator process by calling the EnableMTQViewConstruct function.

If **MTQQuery**=F in the tms1.cfg file on the server where this function is run, EnableMTQViewConstruct can be called to override this value on a single TurboIntegrator process.

You can enable EnableMTQViewConstruct in either the Prolog or Epilog section of a TurboIntegrator process. For efficiency, enable EnableMTQViewConstruct in the first, or very close to the first, statement in the Prolog section of your process.

Example

Use the following TurboIntegrator commands to enable multi-threaded query processing when calculating a view to be used as a TurboIntegrator datasource for a single TurboIntegrator process.

```
EnableMTQViewConstruct()
```

See also “DisableMTQViewConstruct” on page 595.

After enabling EnableMTQViewConstruct in a process, it can only be disabled on the last line in the Epilog section. If you attempt to use disable DisableMTQViewConstruct anywhere else in the process, the process will not compile.

If the mode is enabled in one TurboIntegrator process, it remains enabled until explicitly disabled or until the chore completes. This means you can enable the mode in a process and then run a series of TurboIntegrator processes before disabling it.

ViewColumnDimensionSet

ViewColumnDimensionSet sets a column dimension for a TM1 view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewColumnDimensionSet(CubeName, ViewName, DimName, StackPosition);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the column dimension.

Argument	Description
ViewName	The view for which you are setting the column dimension.
DimName	The dimension you want to set as a column dimension for the view.
StackPosition	A number that indicates the stack position of the dimension in the view. This is a 1-based number. 1 indicates the top-most stack position. 2 indicates a position below 1, and so on.

Example

```
ViewColumnDimensionSet('98sales', 'Quarter1', 'Month',1);
```

This example sets Month as a column dimension for the 1Quarter view of the 98sales cube. In the event of stacked column dimensions, Month is placed in the top-most position.

ViewColumnSuppressZeroesSet
ViewColumnSuppressZeroesSet suppresses or enables the display of columns containing only zero values in a TM1 cube view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewColumnSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of columns containing only zeroes in the view. Specify 0 to enable the display of columns containing only zeroes.

Example

```
ViewColumnSuppressZeroesSet('99sales', '1st QuarterActuals', 1);
```

This example suppresses the display of any columns containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewConstruct
ViewConstruct constructs, pre-calculates, and stores a Stargate view in memory on a server. This function is useful for pre-calculating and storing large views so they can be quickly accessed after a data load or update.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewConstruct(CubeName, ViewName);
```

Argument	Description
CubeName	The cube from which you want to construct the view.
ViewName	The view you want to construct. This view must be an existing public view on the server.

Example

```
Viewconstruct('99sales', '1st Quarter Actuals');
```

This example creates the view `1st Quarter Actuals` of the `99sales` cube.

ViewCreate

ViewCreate creates an empty view of a specified cube.

When the optional `AsTemporary` argument is set to `1`, the view is temporary and persists only for the duration of the TurboIntegrator process or chore in which the view is created.

Note:

For TM1 Server version 11.2.0 and earlier, temporary views were visible and usable only by the process that created it and any of its child processes. Temporary views were not visible to the ancestor and sibling processes. You could create same-named views in sibling child processes with the same parent process.

For TM1 Server version 11.3.0 and later, these temporary views are visible to the ancestor and sibling processes. If a parent TurboIntegrator process A invokes two child TurboIntegrator processes A1 and A2, and the child TurboIntegrator process A1 creates a temporary view S, the temporary view S exists for the duration of the parent TurboIntegrator process A. You cannot create a temporary view with the same name S in the sibling TurboIntegrator process A2 since the view is visible and usable by siblings A1 and A2.

While a temporary view exists, the temporary view takes precedence over any same-named public view. If another TurboIntegrator function references a view that exists in both a temporary and permanent state, the function operates upon the temporary view.

Temporary objects have transaction scope. When a transaction is committed, all temporary objects are cleaned up. If a chore is run in single-commit mode where all processes in the chore are logically run within the context of one transaction, then temporary objects that are created in a process still exist, visible, and available for use, in subsequent processes run by the chore. However, in multi-commit mode, these processes are cleaned up at commit time of the transaction that wrapped the execution of the process that created the temporary object.

There is no locking associated with a temporary view, as a temporary view is never saved. This can result in improved performance, because there is no need for TurboIntegrator to wait for locks to be released before operating upon a temporary view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewCreate(Cube, ViewName, <AsTemporary>);
```

Argument	Description
Cube	The parent cube of the view you are creating.
ViewName	The name you want to assign to the view.
AsTemporary	This is an optional argument that specifies whether the view being created is temporary. 1 indicates a temporary view, 0 indicates a permanent view. If this argument is omitted, the view is permanent.

Example

This example creates a temporary view named 1st Quarter Actuals from the Sales cube.

```
ViewCreate('Sales', '1st Quarter Actuals', 1);
```

ViewCreateByMDX

ViewCreateByMDX creates a view with a specified MDX expression.

When the optional `AsTemporary` argument is set to 1, the view is temporary and persists only for the duration of the TurboIntegrator process or chore in which the view is created.

Note:

For TM1 Server version 11.2.0 and earlier, temporary views were visible and usable only by the process that created it and any of its child processes. Temporary views were not visible to the ancestor and sibling processes. You could create same-named views in sibling child processes with the same parent process.

For TM1 Server version 11.3.0 and later, these temporary views are visible to the ancestor and sibling processes. If a parent TurboIntegrator process A invokes two child TurboIntegrator processes A1 and A2, and the child TurboIntegrator process A1 creates a temporary view S, the temporary view S exists for the duration of the parent TurboIntegrator process A. You cannot create a temporary view with the same name S in the sibling TurboIntegrator process A2 since the view is visible and usable by siblings A1 and A2.

While a temporary view exists, the temporary view takes precedence over any same-named public view. If another TurboIntegrator function references a view that exists in both a temporary and permanent state, the function operates upon the temporary view.

Temporary objects have transaction scope. When a transaction is committed, all temporary objects are cleaned up. If a chore is run in single-commit mode where all processes in the chore are logically run within the context of one transaction, then temporary objects that are created in a process still exist, visible, and available for use, in subsequent processes run by the chore. However, in multi-commit mode, these processes are cleaned up at commit time of the transaction that wrapped the execution of the process that created the temporary object.

There is no locking associated with a temporary view, as a temporary view is never saved. This can result in improved performance because there is no need for TurboIntegrator to wait for locks to be released before operating upon a temporary view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewCreateByMDX(Cube, ViewName, MDX_expression , <AsTemporary>);
```

Argument	Description
Cube	The parent cube of the view you are creating.
ViewName	The name you want to assign to the view.
MDX_expression	A string value containing a valid MDX view expression.
AsTemporary	This is an optional argument that specifies whether the view being created is temporary. 1 indicates a temporary view; 0 indicates a permanent view. If this argument is omitted, the view is permanent.

Example

This example, based on the Planning Sample database, creates a temporary view named Account in the plan_BudgetPlan cube.

```
ViewCreateByMDX('plan_BudgetPlan', 'Account',
    'select {[plan_version].[FY 2003 Budget]} on 0,
    {[plan_business_unit].[10300]} on 1 from plan_budgetplan where
    [plan_department].[200][plan_chart_of_accounts].[41101][plan_exchange_rates].[local]
[plan_source].[goal][plan_time].[Jan-2003]'
    ,1);
```

ViewDestroy

ViewDestroy deletes a view from the TM1 database.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewDestroy(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you are deleting.
ViewName	The name of the view you want to delete.

Example

```
ViewDestroy('99sales', '1st Quarter Actuals');
```

This example deletes the 1st Quarter Actuals view of the 99sales cube.

ViewExists

ViewExists determines whether a specific public view exists on the server from which a TurboIntegrator process is executed. The function returns 1 if the view exists on the server, otherwise it returns 0. Note that this function cannot be used to determine the existence of private views.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExists(CubeName, ViewName);
```

Argument	Description
CubeName	The name of the cube that is the parent of the view whose existence you want to confirm.
ViewName	The name of the public view whose existence you want to confirm

Example

```
ViewExists('Inventory', 'FebClosing');
```

This example determines if FebClosing view of the Inventory cube exists on the server.

ViewExtractFilterByTitlesSet

ViewExtractFilterByTitlesSet sets an option to filter by titles on consolidated values that are excluded from a view or any associated view extracts.

TM1 allows the storing of strings on calculated values. When you exclude a calculated value from a view or view extract you may want to exclude the message string also from the view.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExtractFilterByTitlesSet (Cube, ViewName, FilterByTitles, Temporary);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.
ViewName	The view for which you are setting the option.
FilterByTitles	A binary value that turns the option on or off. Specify 0 to include titles stored on consolidated values. This is the current and default behavior. Specify 1 to exclude titles stored on consolidated values.
Temporary	A Boolean value that indicates whether the settings are temporary.

Example

```
ViewExtractFilterByTitlesSet ('99sales', '1st QuarterActuals', 1, 0);
```

ViewExtractSkipCalcsSet

ViewExtractSkipCalcsSet sets an option to include/exclude consolidated values in a view **and** any associated view extracts. A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

ViewExtractSkipCalcsSet is the equivalent of the Skip Consolidated Values option in the View Extract dialog box.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExtractSkipCalcsSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.
ViewName	The view for which you are setting the option.
Flag	A binary value that turns the option on or off. Specify 1 to exclude consolidated values from the view extract. Specify 0 to include consolidated values.

Example

```
ViewExtractSkipCalcsSet ('99sales', '1st Quarter Actuals',1);
```

This example turns on the Skip Consolidated Values option for the 1st Quarter Actuals view. The view extract will not include any consolidated values.

Note about the impact of enabling a specific combination of view manipulation functions

Consider the scenario when all of these conditions are true:

- the measure is a string
- ViewExtractSkipCalcsSet = 1
- ViewExtractSkipConsolidatedStringsSet = 0 (function is not used)
- ViewExtractSkipRuleValuesSet = 0 (function is not used)

In this scenario, the output is different, depending on whether you enable the ViewExtractSkipZeroesSet function.

- If you set ViewExtractSkipZeroesSet = 0, the TM1 server enumerates every possible cube cell, not just the existing data cells. This situation is rather unusual, since enumerating all possible cells means that the number of cells scanned is the product of the sizes of all of the dimensions of the cube. This product can quickly become very large. In this mode, the ViewExtractSkipCalcsSet function skips all consolidated cells, even if the measure is a string.
- If you set ViewExtractSkipZeroesSet = 1, the TM1 server scans only the cells actually in the cube. In this mode, a string stored on a consolidated cell is treated as a simple leaf (the cell after all has a simple value and is a leaf). Therefore, even though the ViewExtractSkipCalcsSet function is enabled, the entry is not skipped since this cell is not a calculated consolidated cell. In this case, if you want the entries to be skipped, you must enable the ViewExtractSkipConsolidatedStringsSet function.

ViewExtractSkipConsolidatedStringsSet

ViewExtractSkipConsolidatedStringsSet sets an option to exclude strings on consolidated values that are excluded from a view or any associated view extracts. A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

TM1 allows the storing of strings on calculated values. When you exclude a calculated value from a view or view extract you may want to exclude the message string also from the view.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExtractSkipConsolidatedStringsSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.
ViewName	The view for which you are setting the option.
Flag	A binary value that turns the option on or off. Specify 0 to include strings stored on consolidated values. This is the current and default behavior. Specify 1 to exclude strings stored on consolidated values.

Note: Read about the impact of enabling a specific combination of view manipulation functions.

Example

```
ViewExtractSkipConsolidatedStringsSet ('99sales', '1st QuarterActuals', 1);
```

This example turns on the Skip Rule for Consolidated String option for the extract created from the 1st Quarter Actuals view. The extract will not include any string on the consolidated value.

ViewExtractSkipRuleValuesSet

ViewExtractSkipRuleValuesSet sets an option to include/exclude rule-calculated values in a view **and** any associated view extracts. A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

ViewExtractSkipRuleValuesSet is the equivalent of the Skip Rule Calculated Values option in the View Extract dialog box.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExtractSkipRuleValuesSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.

Argument	Description
ViewName	The view for which you are setting the option.
Flag	A binary value that turns the option on or off. Specify 1 to exclude rule-calculated values from the extract. Specify 0 to include rule-calculated values.

Note: Read about the impact of enabling a specific combination of view manipulation functions.

Example

```
ViewExtractSkipRuleValuesSet ('99sales', '1st QuarterActuals', 1);
```

This example turns on the Skip Rule Calculated Values option for the extract created from the 1st Quarter Actuals view. The extract will not include any rule-calculated values.

ViewExtractSkipZeroesSet

ViewExtractSkipZeroesSet sets an option to include/exclude zero values in a view **and** any associated view extracts. A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

ViewExtractSkipZeroesSet is the equivalent of the Skip Zero/Blank Values option in the View Extract dialog box.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewExtractSkipZeroesSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the Skip Zeroes option.
ViewName	The view for which you are setting the Skip Zeroes option.
Flag	A binary value that turns the option on or off. Specify 1 to exclude zeroes from the extract. Specify 0 to include zeros. When UNDEFVALS is used to represent zeroes, the values are not excluded when the Flag argument is 1.

Note: Read about the impact of enabling a specific combination of view manipulation functions.

Example

```
ViewExtractSkipZeroesSet ('99sales', '1st Quarter Actuals',1);
```

This example turns on the Skip Zeroes option for the extract created from the 1st Quarter Actuals view. The extract will not include any zero or blank values.

ViewMDXSet

ViewMDXSet sets the MDX expression for an existing MDX view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewMDXSet(Cube, ViewName, MDX_expression);
```

Argument	Description
Cube	The parent cube of the view you are creating.
ViewName	The name you want to assign to the view.
MDX_expression	A string value containing a valid MDX view expression.

Example

```
ViewMDXSet('Sales', 'Account',
    "select {[plan_version].[FY 2003 Budget]} on 0,
    {[plan_business_unit].[10300]} on 1 from plan_budgetplan where
    [plan_department].[200][plan_chart_of_accounts].[41101][plan_exchange_rates].[local]
[plan_source].[goal][plan_time].[Jan-2003]"
    );
```

This example sets the MDX expression for the "Account" view from the "Sales" cube.

ViewMDXGet

ViewMDXGet retrieves the MDX expression for an existing MDX view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewMDXGet(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you are creating.
ViewName	The name you want to assign to the view.

Example

```
ViewMDXGet('Sales', 'Account');
```

This example retrieves the MDX expression from the "Account" view.

ViewRowDimensionSet

ViewRowDimensionSet sets a row dimension for a TM1 view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewRowDimensionSet(CubeName, ViewName, DimName, StackPosition);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the row dimension.
ViewName	The view for which you are setting the row dimension.
DimName	The dimension you want to set as a row dimension for the view.
StackPosition	A number that indicates the stack position of the dimension in the view. This is a 1-based number. 1 indicates the left-most stack position. 2 indicates a position to the right of 1, and so on. **Note:** It is possible for a TM1 client to set a Tm1p.ini parameter (BrowseDisplayReadsRightToLeft=T) that reverses the orientation of data in the Cube Viewer. When the orientation of data is reversed, the stack positions are also reversed. 1 indicates the right-most stack position. 2 indicates a position to the left of 1, and so on.

Example

```
ViewRowDimensionSet('98sales', 'Quarter1', 'Month',1)
```

This example sets Month as a row dimension for the 1Quarter view of the 98sales cube. In the event of stacked row dimensions, Month is placed in the left-most position.

ViewRowSuppressZeroesSet

ViewRowSuppressZeroesSet suppresses or enables the display of rows containing only zero values in a TM1 cube view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewRowSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.

Argument	Description
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of rows containing only zeroes in the view. Specify 0 to enable the display of rows containing only zeroes.

Example

```
ViewRowSuppressZeroesSet('99sales', '1st Quarter Actuals',1);
```

This example suppresses the display of any rows containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewSubsetAssign

ViewSubsetAssign assigns a named subset to a cube view.

Note: It is possible to create a temporary subset with the CreateSubset or CreateSubsetByMDX functions. If you attempt to use ViewSubsetAssign to assign a temporary subset to a permanent view, the function will fail with error notification.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewSubsetAssign(Cube, ViewName, DimName, SubName);
```

Argument	Description
Cube	The parent cube of the view to which you are assigning a subset.
ViewName	The view to which you are assigning a subset.
DimName	The parent dimension of the subset you are assigning to the view.
SubName	The name of the subset you want to assign to the view.

Example

```
ViewSubsetAssign('99sales', '1st Quarter Actuals', 'Month','Q1');
```

This example assigns the Q1 subset of the Month dimension to the 1st Quarter view.

ViewSuppressZeroesSet

ViewSuppressZeroesSet suppresses or enables the display of all rows and columns containing only zero values in a TM1 cube view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of rows or columns containing only zeroes in the view. Specify 0 to enable the display of rows and columns containing only zeroes.

Example

```
ViewSuppressZeroesSet('99sales', '1st Quarter Actuals',1);
```

This example suppresses the display of any rows or columns containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewTitleDimensionSet

ViewTitleDimensionSet sets a title dimension for a TM1 view.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewTitleDimensionSet(CubeName, ViewName, DimName);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the title dimension.
ViewName	The view for which you are setting the title dimension.
DimName	The dimension you want to set as a title dimension for the view.

Example

```
ViewTitleDimensionSet('98sales', 'Quarter1', 'Month');
```

This example sets Month as a title dimension for the 1Quarter view of the 98sales cube.

ViewTitleElementSet

ViewTitleElementSet sets a title element for a TM1 view. ViewTitleElementSet is used in conjunction with the ViewTitleDimensionSet function.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ViewTitleElementSet(CubeName, ViewName, DimName, Index);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the title element.
ViewName	The view for which you are setting the title element.
DimName	The parent dimension of the title element.
Index	An index into the specified dimension that indicates the element to be set as the title element.

Example

```
ViewTitleElementSet('98sales', 'Quarter1', 'Model',3);
```

This example sets the third element of the Model dimension as a title element for the Quarter1 view of the 98sales cube.

ViewZeroOut

ViewZeroOut sets all data points in a view to zero.

This function is valid in TM1 TurboIntegrator processes only.

Note: When using ViewZeroOut on a cube which has UNDEFVALS enabled, the values in the view will be set to zero, not the UNDEFVAL state.

Syntax

```
ViewZeroOut(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you want to zero out.
ViewName	The view you want to zero out.

Example

```
ViewZeroOut('99sales', '1st Quarter Actuals');
```

This example sets all data points in the 1st Quarter Actuals view to zero.

Miscellaneous TurboIntegrator Functions

These functions facilitate miscellaneous tasks.

AddInfoCubeRestriction

AddInfoCubeRestriction filters InfoCube data as it is pulled into TM1. Use this function to restrict the values that are imported for a specified characteristic. This function must be placed in the Prolog. The function can be called multiple times to filter more than one characteristic in a single process.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
AddInfoCubeRestriction(STRING CharactName, STRING sign,STRING compOperator,
STRING lowValue, STRING highValue)
```

Argument	Description
STRING CharactName	Contains the technical name of the characteristic to be restricted. The data type has to be a character string with a length equal to or less than 30.
STRING sign	Contains either I (= inclusive) or E (= exclusive). Exclusive is the logical NOT for the restriction specified by this row. The data type has to be a character of length 1.
STRING compOperator	Contains the relational comparative operator. The data type has to be a character string of length 2. Valid comparative operators are: 'EQ' = equal 'NE' = not equal 'LT' = less than 'GT' = grater than 'LE' = less or equal 'GE' = grater or equal 'BT' = between 'NB' = not between
STRING lowValue	Contains the low value for the operator specified in the row before. The data type has to be a character string with a length equal to or less than 60.
STRING highValue	Contains the high value for the operator specified two rows before. The data type has to be a character string with a length equal to or less than 60. It is only needed for the operators BT and NB, otherwise it is ignored, and in this case an empty string should be placed here.

Example

The following example returns all characteristic values between 1997 and 2000.

```
AddInfoCubeRestriction('0CALYEAR','E','BT','1997','2000');
```

The following example returns all characteristic values not between 1997 and 2000.

```
AddInfoCubeRestriction('0CALYEAR','I','NB','1997', '2000') ;
```

The following example returns all characteristic values not equal to USD.

```
AddInfoCubeRestriction('0DOC_CURRCY', 'I', 'NE', 'USD','') ;
```

ExecuteJavaN

ExecuteJavaN executes a Java™ TurboIntegrator process that returns a number. If you want to execute a Java TurboIntegrator process that returns a string, use ExecuteJavaS.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ExecuteJavaN('JavaTIClass', ['OptionalParameter1', 'OptionalParameter2', ...] )
```

Argument	Description
JavaTIClass	The fully qualified name of the Java TurboIntegrator class you want to execute.
OptionalParameters	Optional parameters that are passed to the Java TurboIntegrator process itself. You can pass as many parameters as necessary, including none. You can pass only strings as parameters, you cannot pass numbers. You can use the StringToNumber TurboIntegrator function to pass numbers to Java TurboIntegrator scripts. For more information, see "StringToNumber" in the *TM1 Reference*.

A Java TurboIntegrator class, which returns a number and can be called from ExecuteJavaN, must use the following pattern:

```
package com.example;

import com.ibm.cognos.tm1.javati.JavaTI;

@JavaTI
public class MyTestTI {
    public static double MyTestTI (String [] args) [
    ...
    return ...;
    }
}
```

Example

```
ExecuteJavaN('com.example.MyTestTI', 'First', 'Second', 'Third');
```

ExecuteJavaS

ExecuteJavaS executes a Java TurboIntegrator process that returns a string. If you want to execute a Java TurboIntegrator process that returns a number, use ExecuteJavaN.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
ExecuteJavaS('JavaTIClass', ['OptionalParameter1', 'OptionalParameter2', ...] )
```

Argument	Description
JavaTIClass	The fully qualified name of the Java TurboIntegrator class you want to execute.

Argument	Description
OptionalParameters	Optional parameters that are passed to the Java TurboIntegrator process itself. You can pass as many parameters as necessary, including none. You can pass only strings as parameters, you cannot pass numbers. You can use the StringToNumber TurboIntegrator function to pass numbers to Java TurboIntegrator scripts. For more information, see "StringToNumber" in the *TM1 Reference*.

A Java TurboIntegrator class, which returns a string and is called from ExecuteJavaS, must use the following pattern.

```
package com.example;

import com.ibm.cognos.tm1.javati.JavaTI;

@JavaTI
public class MyTestTI {
    public static String MyTestTI (String [] args) [
    ...
    return ...;
    }
}
```

Example

```
ExecuteJavaS('com.example.MyTestTI', 'First', 'Second', 'Third');
```

Expand

Expand expands TurboIntegrator variable names, enclosed in % signs, to their values at run time. A common use of the Expand function is to pass the value of TurboIntegrator variables to the ODBCOutput function.

If the variable name represents a string variable, the entire variable expression must be enclosed on quotes. For example, "%V1%".

If Expand is fed with a numerical value, an implicit type conversion is performed and the numerical value is converted into a string.

That string has a fixed minimum length of 10 characters. If the converted number is too small to fill 10 characters, it is padded with leading spaces. Only three leading decimal characters are converted. For example, a numerical value of 0.123456789 is converted into the string "0.123".

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
Expand(String);
```

Argument	Description
String	Any string that includes TurboIntegrator variable names enclosed in % signs.

Example

```
ODBCOutPut( 'TransData', Expand( 'INSERT INTO SALES( MONTH, PRODUCT, SALES )
VALUES ( "%V0%", "%V1%",%V2% )' ) );
```

This example illustrates the use of the Expand function within the ODBCOutput function. The example inserts records into a relational table named Sales that consists of three columns: Month, Product, and Sales.

The Expand function converts the variables V0, V1, and V2 to their actual values within the view. Assuming that the first value in the view is 123.456, and is defined by the elements Jan and Widget

```
Expand( 'INSERT INTO SALES ( MONTH, PRODUCT, SALES ) VALUES ("%V0%", "%V1%",%V2% )' )
```

becomes

```
'INSERTINTO SALES ( MONTH, PRODUCT, SALES ) VALUES ( Jan, Widget,123.456 )'
```

at run time.

FileExists

FileExists determines whether a specified file exists. The function returns 1 if the file exists, 0 if it does not.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
FileExists(File);
```

Argument	Description
File	The name of a file. If a full parth is not specified, TM1 searches for the file in the server data directory.

Example

```
FileExists('C:\tm1s7\pdata\model.dim');
```

This example determines if the model.dim file exists.

LogOutput

LogOutput writes a message to the `tm1server.log` file when an error of a specified severity level is encountered in a TurboIntegrator process.

This function is valid in TM1 TurboIntegrator processes only.

Prerequisite

To enable message logging from TurboIntegrator, you must add the `TM1.TILogOutput` debugger to the `tm1-log.properties` file and set the debugger to the desired level. For example, adding `TM1.TILogOutput=DEBUG` to `tm1-log.properties` enables logging for all severity levels. For details on the `tm1-log.properties` file, see "Configuring and Enabling Server Message Logging" in *TM1 Operations*.

Syntax

```
LogOutput('SeverityLevel', 'MessageString');
```

Argument	Description
SeverityLevel	The severity level that initiates logging to the tm1server.log file. Valid values for this argument are: • 'DEBUG' • 'INFO' • 'WARN' • 'ERROR' • 'FATAL'
MessageString	The message you want to write to the tm1server.log file. The message string can be a string enclosed in single quotes or can be another TurboIntegrator function that returns a string.

Examples

```
LogOutput('WARN', 'TI process encountered a warning condition');
```

```
LogOutput('ERROR', TM1User() );
```

TM1User

TM1User returns a string giving the current TM1 client. When executed in a process that the user is running directly, it will return the user's TM1 client name. When executed in a chore that the user runs directly, it will also return the user's TM1 client name.

If run from a scheduled chore, it will return a name in the form *R*<chore name>*, for example, `R*UpdateRegionDimension`.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
TM1User()
```

WildcardFileSearch

WildcardFileSearch lets you use wildcard characters to search for files in a specified directory.

The results of the WildCardFileSearch function may vary depending on the operating system in use. Files in a Windows directory are sorted in alphabetical order while files in a UNIX directory are sorted in random order. Because the order of sorting varies between the operating systems, the identical WildCardFileSearch function executed against identical directories, one on Windows and one on UNIX, will yield different results.

This function is valid in TM1 TurboIntegrator processes only.

Syntax

```
WildcardFileSearch( Pathname, PriorFilename);
```

Argument	Description
Pathname	A pathname to files for which you want to search. The pathname must end in a filename, which can contain a wildcard sequence using the * and/or ? characters. The ? wildcard character matches any single character. The * wildcard character matches zero or more characters.
PriorFilename	The name of an existing file in the specified directory. This filename cannot contain wildcard characters. The wildcard search specified by the Pathname argument will commence *AFTER* this file. If you pass an empty string as the PriorFilename argument, the WildcardFileSearch function returns the first file that matches the wildcard sequence specified by the Pathname argument.

Example

The following example shows the use of the WildcardFileSearch function to determine the first server log file generated in 2004:

```
file = WildcardFileSearch( 'C:\Program Files\Cognos\TM1\Custom\
TM1Data\SData\tm1s2004*.log', ' ');
```

This example returns the first file matching the wildcard sequence 'tm1s2004*.log' from the C:\Program Files\Cognos\TM1\Custom\TM1Data\SData\ directory.

Because server log files are named and saved with sequential time stamps, and because the second parameter to WildcardFileSearch is empty, the function returns the first server log file starting with the characters 'tm1s2004'. This would be the first server log file generated in the year 2004.

The following example shows the use of the WildcardFileSearch function to return the first server log file generated after tm1s20040211153827.log was generated:

```
file = WildcardFileSearch( 'C:\Program Files\Cognos\TM1\Custom\
TM1Data\SData\tm1s*.log', 'tm1s20040211153827.log
');
```

This example begins searching the C:\Program Files\Cognos\TM1\Custom\TM1Data\SData\ directory immediately after the tm1s20040211153827.log file, and returns the first subsequent file matching the 'tm1s*.log' wildcard sequence.

tm1s20040220175522.log is the first file that occurs after tm1s20040211153827.log and matches the wildcard sequence. Accordingly, the example returns tm1s20040220175522.log.

TurboIntegrator Variables

IBM Planning Analytics variables are listed here by categories.

TurboIntegrator Local Variables

When you execute a TurboIntegrator process, a set of implicit local variables is generated. Local variables exist only in the context of the process in which they are used, and are not available outside of the process. Local variables are destroyed when a process exits. These variables, listed below, can be overwritten to manipulate a process.

DatasourceASCIIDecimalSeparator
This TurboIntegrator local variable sets the decimal separator to be used in any conversion of a string to a number or a number to a string. If you set this variable you must also set the DatasourceASCIIThousandSeparator variable.

The character specified must be a standard ASCII printable character, with a decimal value between 33 and 127 inclusive.

Syntax

```
DatasourceASCIIDecimalSeparator='Char';
```

or

```
DatasourceASCIIDecimalSeparator=Char(xx);
```

Argument	Description
Char	The ASCII character to be used as a separator. The character can be specified as a character enclosed in single quotes, or as an ASCII Char decimal code without quotes.

Either of the following examples sets the comma character (,) as the separator.

```
DatasourceASCIIDecimalSeparator=',';
```

```
DatasourceASCIIDecimalSeparator=Char(44);
```

DatasourceASCIIDelimiter
This TurboIntegrator local variable sets the ASCII character to be used as a field delimiter when the DatasourceType is 'CHARACTERDELIMITED".

The character specified must be a standard ASCII printable character, with a decimal value between 33 and 127 inclusive.

Syntax

```
DatasourceASCIIDelimiter='Char';
```

or

```
DatasourceASCIIDelimiter=Char(xx);
```

Argument	Description
Char	The ASCII character to be used as a delimiter. The character can be specified as a character enclosed in single quotes, or as an ASCII Char decimal code without quotes.

Either of the following examples sets the hyphen character (-) as the field delimiter.

```
DatasourceASCIIDelimiter='-';
```

```
DatasourceASCIIDelimiter=Char(45);
```

DatasourceASCIIHeaderRecords

This TurboIntegrator local variable indicates the number of records to be skipped before processing the data source.

Syntax

```
DatasourceASCIIHeaderRecords=N;
```

Argument	Description
N	The number of records to be skipped before processing the data source.

DatasourceASCIIQuoteCharacter

This TurboIntegrator local variable sets the ASCII character used to enclose the fields of the source file when DatasourceType is 'CHARACTERDELIMITED'.

The character specified must be a standard ASCII printable character, with a decimal value between 33 and 127 inclusive.

Syntax

```
DatasourceASCIIQuoteCharacter='Char';
```

or

```
DatasourceASCIIQuoteCharacter=Char(xx);
```

Argument	Description
Char	The ASCII character that encloses fields in the data source. The character can be specified as a character enclosed in single quotes, or as an ASCII Char decimal code without quotes.

Either of the following examples sets the asterisk character (*) as the field delimiter.

```
DatasourceASCIIQuoteCharacter='*';
```

```
DatasourceASCIIQuoteCharacter=Char(42);
```

DatasourceASCIIThousandSeparator

This TurboIntegrator local variable sets the thousands separator to be used in any conversion of a string to a number or a number to a string.

If you set this variable you must also set the DatasourceASCIIDecimalSeparator variable.

The character specified must be a standard ASCII printable character, with a decimal value between 33 and 127 inclusive.

Syntax

```
DatasourceASCIIThousandSeparator='Char';
```

or

```
DatasourceASCIIThousandSeparator=Char(xx);
```

Argument	Description
Char	The ASCII character to be used as a separator. The character can be specified as a character enclosed in single quotes, or as an ASCII Char decimal code without quotes.

Either of the following examples sets the period character (.) as the thousands separator.

```
DatasourceASCIIThousandSeparator='.';
```

```
DatasourceASCIIThousandSeparator=Char(46);
```

DatasourceCubeview

This TurboIntegrator local variable sets the view to process if the DatasourceType is 'VIEW'.

Syntax

```
DatasourceCubeview='ViewName';
```

Argument	Description
ViewName	The name of the view to be processed. This must be an existing view of the cube specified by the DataSourceNameForServer variable.

DatasourceDimensionSubset

This TurboIntegrator local variable sets the subset to process if the DatasourceType is 'SUBSET.'

DatasourceNameForServer=*Dimension name* is also needed in conjunction with DATASOURCEDIMENSIONSUBSET so TM1 can identify where the subset is located.

Syntax

```
DatasourceDimensionSubset='SubsetName';
```

Argument	Description
SubsetName	The name of the subset to be processed.

DatasourceNameForServer

This TurboIntegrator local variable sets the name of the data source (.cma file, cube name, ODBC source) used by the server when executing the process.

Syntax

```
DatasourceNameForServer='Name';
```

Argument	Description
Name	For a .cma data source, the full path of the .cma file. For cubes, the cube name prefaced with the string 'local:'. For an ODBC source, the source name.

DatasourceNameForClient

This TurboIntegrator local variable sets the name of the data source (.cma file, cube name, ODBC source) used by the client when creating or editing the process.

Syntax

```
DatasourceNameForClient='Name';
```

Argument	Description
Name	For a .cma data source, the full path of the .cma file. For cubes, the cube name prefaced with the string 'local:'. For an ODBC source, the source name.

DataSourceODBOCatalog

This TurboIntegrator local variable sets the name of the database collection that contains the cubes, dimensions or other objects to which you want to connect. For Microsoft Analysis Services, this is the name of the database.

Syntax

```
DataSourceODBOCatalog='Catalog';
```

Argument	Description
Catalog	The name of the database collection to which you want to connect.

DataSourceODBOConnectionString

This TurboIntegrator local variable sets any additional connection parameters that may be required to connect to the OLAP server.

Syntax

```
DataSourceODBOConnectionString='String';
```

Argument	Description
String	The value used to define additional connection parameters. Assign these parameters to this variable, delimited by semi-colons.

DataSourceODBOCubeName

This TurboIntegrator local variable sets the name of the cube from the OLAP server that you want to use as a data source.

Syntax

```
DataSourceODBOCubeName='Name';
```

Argument	Description
Name	The name of the cube to be used.

DataSourceODBOHierarchyName

This TurboIntegrator local variable sets the name of the hierarchy for the specific dimension you are using as a data source. You use this variable for other OLAP products, such as SAP BW, where a hierarchy is a separate object.

This variable is not used with TM1 data sources.

Syntax

```
DataSourceODBOHierarchyName='Name';
```

Argument	Description
Name	The name of the hierarchy for a specific dimension.

DataSourceODBOLocation

This TurboIntegrator local variable sets the name of the location (system) where the OLAP server is running.

TM1 uses this variable, but other OLAP servers do not. For TM1, this is the location where the Admin Host is running.

Syntax

```
DataSourceODBOLocation='Location';
```

Argument	Description
Location	The name of the location (system) for the OLAP server.

DataSourceODBOProvider

This TurboIntegrator local variable sets the name of the ODBO provider that you want to use as a data source. This is the full name that is assigned by the ODBO provider manufacturer to identify their multidimensional database server.

You must use the name of an ODBO provider that is installed on your server.

Syntax

```
DataSourceODBOProvider='Provider';
```

Argument	Description
Provider	The name of the ODBO provider to use as a data source. Commonly-used provider names include: TM1 OLE DB MD Provider Microsoft OLE DB Provider for OLAP Services 8.0 SAP BW OLE DB Provider

DataSourceODBOSAPClientID

This TurboIntegrator local variable sets the client number that corresponds to the UI version on the SAP server to which you want to connect.

Syntax

```
DataSourceODBOSAPClientID='ID';
```

Argument	Description
ID	A number that corresponds to the UI version on the SAP server. For example, 498.

DataSourceODBOSAPClientLanguage

This TurboIntegrator local variable sets the language specification for the language of the SAP system to which you want to connect.

Syntax

```
DataSourceODBOSAPClientLanguage='Language';
```

Argument	Description
Language	The language specification of the SAP system. For US English, use EN. For German, use DE. For other languages, refer to your SAP documentation.

DatasourcePassword

This TurboIntegrator local variable sets the password used to connect to the data source.

Syntax

```
DatasourcePassword='Password';
```

Argument	Description
Password	The password used to connect to the data source set with DatasourceNameForServer.

DatasourceQuery

This TurboIntegrator local variable sets the query string to use with the data source.

Syntax

```
DatasourceQuery='Query';
```

Argument	Description
Query	The query string to use with the data source that was set with DatasourceNameForServer.

DataSourceSAPUsingRoleAuths

The DataSourceSAPUsingRoleAuths local variable instructs the TurboIntegrator process to ignore security information when processing an SAP datasource. This variable must be placed in the Prolog.

Syntax

```
DataSourceSAPUsingRoleAuths='0'
```

Argument	Description
0	Security information is *ignored* when processing an SAP datasource.
1	Security information is *read* when processing an SAP datasource.

DataSourceSAPUsingTexts

The DataSourceSAPUsingTexts local variable instructs the TurboIntegrator process to ignore characteristic descriptions when processing an SAP datasource, resulting in a decreased memory consumption and increased performance. This variable must be placed in the Prolog.

Syntax

```
DataSourceSAPUsingTexts='0'
```

Argument	Description
0	Characteristic descriptions *are ignored* when processing an SAP datasource. The characteristic technical name is imported into TM1 as both an element name and alias.
1	Characteristic descriptions *are read* when processing an SAP datasource.

DatasourceType

This TurboIntegrator local variable sets the type of the data source.

Syntax

```
DataSourceType='Type';
```

Argument	Description
Type	Valid types include: 'CHARACTERDELIMITED', 'POSITIONDELIMITED', 'VIEW', 'SUBSET', ODBC' and 'OLEDBOLAP', 'SAPCHARACTERISTICTEXTS'

DatasourceUsername

This TurboIntegrator local variable sets the name used to connect to the data source.

Syntax

```
DatasourceUserName='Name';
```

Argument	Description
Name	The name used to connect to the data source set with DatasourceNameForServer.

MinorErrorLogMax

This TurboIntegrator local variable defines the number of minor errors that will be written to the TM1ProcessError.log file during process execution. If this variable is not defined in the process, the default number of minor errors written to the log file is 1000.

Syntax

```
MinorErrorLogMax=N;
```

Argument	Description
N	Value indicating the number of errors that should be written to the log file. Specify an integer greater than zero to set the maximum number of errors written to the log file. Specify a value of 0 to log no errors during process execution. Specify a value of -1 to allow an unlimited number of minor errors to be written to the log file.

```
The following table provides an example error log message
and the corresponding result.
```

Example	Result
MinorErrorLogMax=750;	The log file will accept up to 750 errors.
MinorErrorLogMax=0;	No errors will be written to the log file.
MinorErrorLogMax=-1;	No limit on the number of errors written to the log file.

NValue

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines the value of the current cell when Value_Is_String is 0. (That is, when the current cell is numeric.)

Syntax

```
Nvalue=N;
```

Argument	Description
N	The value of the current cell.

OnMinorErrorDoItemSkip

This TurboIntegrator local variable instructs TurboIntegrator to skip to the next record when a minor error is encountered while processing a record.

This variable is useful in scenarios where a single bad field/value in a record causes multiple minor errors.

For example, if you have 100 CELLPUTN functions in a process and one of the fields in a given record is 'bad' or invalid, the minor error count is incremented by 100. (1 for each CELLPUTN function that encounters the error.) These 100 minor errors count towards the minor error limit defined by MinorErrorLogMax. A TurboIntegrator process fails when it surpasses the number of minor errors defined by MinorErrorLogMax.

If OnMinorErrorDoItemSkip=1; is included in the Prolog tab of the process, the process immediately skips to the next record when a 'bad' or invalid field is encountered in a source record. Using the above example, this results in the minor error count being incremented by just 1, rather than 100.

Syntax

```
OnMinorErrorDoItemSkip=N;
```

Argument	Description
N	Value indicating if item should be skipped when a minor error is encountered. 1 (or any other non-zero value) dictates that the process should skip to the next record when a minor error is encountered. 0 indicates that TurboIntegrator should continue processing the current record when a minor error occurs.

SValue

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines the value of the current cell when Value_Is_String is not 0. (That is, when the current cell contains a string.)

Syntax

```
Svalue='String';
```

Argument	Description
String	The value of the current cell.

TM1ProcessError.log file

When a TurboIntegrator process encounters an error, it generates a TM1ProcessError.log file. This log file is saved to the data directory of the server on which the process resides.

Note: In a Planning Analytics on Cloud environment, the TM1ProcessError.log file is retained for three months. Any TM1ProcessError.log files that are older than three months are permanently deleted during the regularly scheduled maintenance window. If you want retain your TM1ProcessError.log files beyond the three month maintenance interval, please compress them to a zip file. For more information on log file retention in Planning Analytics on Cloud, see https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_cloud_mg.2.0.0.doc/c_paw_log_retention_periods.html.

A TM1ProcessError.log file contains a list of errors that are encountered by the process. For each error encountered, the log file records the tab and line that caused the error, along with a brief description of the error.

When a process error log file is generated, TM1 assigns a unique name that lets you readily identify which TurboIntegrator process generated the error file and the time at which the file was created. File names are assigned with the following convention:

TM1ProcessError_<*time stamp*>_<*process name*>.log.

In this convention, <*time stamp*> is the time (expressed as yyyymmddhhmmss GMT) at which the file was generated and <*process name*> is the name of the TurboIntegrator process that caused the errors.

For example, an error file named TM1ProcessError_20040224203148_ CreateSalesCube.log indicates that the error file was generated at 20:31:48 GMT on February 24, 2004 and that it contains errors that are caused by the CreateSalesCube process.

Manual intervention is required to delete or archive these log files. A new log file is generated each time a TurboIntegrator process has an error (1 log file per TurboIntegrator execution).

Many TurboIntegrator process error logs might be generated for TurboIntegrator processes that run frequently and generate an error on each execution. Many log files generated in the TM1 database log directory might impact performance of the TM1 database when it creates or updates log files.

The Planning Analytics Administration ability to download log files might be impacted by a large number of files in the TM1 database log directory. It's recommended to limit the number of files in the TM1 database logging directory to under 2000 files.

Value_Is_String

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines whether the current cell should be treated as a string or a numeric value.

Syntax

```
Value_Is_String=N;
```

Argument	Description
N	Value indicating if the current cell is a string or a numeric value. 0 dictates that the cell is a number; anything else means the cell is treated as a string.

TurboIntegrator Global Variables

This type of TurboIntegrator variable is associated with an individual TM1 chore or with an individual process and any attendant sub-processes. There are two types of global variables: implicit and user-defined. Implicit global variables are described here. User-defined global variables are described below.

Global variables can be used in two ways:

- Global variables can be declared within a process that is part of a given chore. Once declared, the global variables are available to all other processes that are part of the chore. The variables persist while the chore is executing and for the duration of the current server session. Global variables are destroyed upon server shutdown.
- Global variables can be declared in one process and be made available to any subsequent processes called by the ExecuteProcess( ) function. These sub-processes must use the same global variable declaration statements (described below) to access the global variables.

In the event that a global variable name is identical to a local variable name, the local variable definition takes precedence and overrides the global variable.

Global variables are declared in a TurboIntegrator process using one of the following two functions

NumericGlobalVariable('VariableName');
Use this function to declare a numeric global variable.

StringGlobalVariable('VariableName');
Use this function to define a string global variable.

Implicit Global Variables

When you execute a TurboIntegrator process, a set of implicit global variables is generated. If the process generating the variables is part of a chore, these global variables are available to and can be shared by all other processes within the chore.

In addition, all implicit global variables in a process are available to and can be shared by any subsequent processes called by the ExecuteProcess( ) function.

Though implicit variables are generated by the TurboIntegrator process, you must declare a variable before it can be used in a process

Implicit global variables are declared in a TurboIntegrator process using the NumericGlobalVariable('VariableName');:

Click a link below for details on a specific implicit global variables.

- DataMinorErrorCount.
- MetadataMinorErrorCount.
- ProcessReturnCode.
- PrologMinorErrorCount.

For example, to use the PrologMinorErrorCount implicit global variable in a process, you must first declare the variable as follows:

```
NumericGlobalVariable('PrologMinorErrorCount');
```

DataMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Data portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
DataMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Data portion of the process.

MetadataMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Metadata portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
MetadataMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Metadata portion of the process.

ProcessReturnCode

This TurboIntegrator global variable stores the exit status of the most recently executed TurboIntegrator process.

Syntax

```
ProcessReturnCode=StatusCode;
```

Status Code	Description
ProcessExitByBreak()	Indicates that the process exited because it encountered a ProcessBreak function.
ProcessExitByChoreQuit()	Indicates that the process exited due to execution of the ChoreQuit function.
ProcessExitByChoreRollback()	Indicates that the process exited because it encountered a ChoreRollback function.
ProcessExitByProcessRollback()	Indicates that the process exited because it encountered a ProcessRollback function.
ProcessExitByQuit()	Indicates that the process exited because of an explicit `quit` command.
ProcessExitMinorError()	Indicates that the process executed successfully but encountered minor errors.
ProcessExitNormal()	Indicates that the process executed normally.
ProcessExitOnInit()	Indicates that the process aborted during initialization.
ProcessExitSeriousError()	Indicates that the process exited because of a serious error.
ProcessExitWithMessage()	Indicates that the process exited normally, with a message written to `tm1server.log`.

PrologMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Prolog portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
PrologMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Prolog.

TurboIntegrator User Variables

This type of variable is associated with an individual TM1 user, not with any particular process or chore. User variables can be manipulated from within any TurboIntegrator process or chore while the user with which the variable is associated is logged on to the server.

User variables must be explicitly declared. Once declared, user variables persist for the life of the user's TM1 session (until the user logs off or is disconnected from the server).

User variables are declared in a TurboIntegrator process using one of the following two functions:

- NumericGlobalVariable('VariableName');. Use this function to declare a numeric user variable.
- StringGlobalVariable('VariableName');. Use this function to define a string user variable.

User variables are created the first time such a declaration is encountered in any running TurboIntegrator process.

Once created, the variable name may be referenced and used just like any local or global variable, expect that the variable value persists across processes and chores only for as long as the user who created the variable is logged on to the server.

MDX Function Support

All TM1 supported Microsoft-defined and TM1 specific functions are listed in this section.

Support for Microsoft-defined MDX Expressions and Functions

TM1 supports Microsoft-defined MDX expressions and functions. The TM1 implementation of these functions and expressions is based on the definitions in the Microsoft MSDN library, which is available at the Microsoft MSDN website.

Supported Member Expressions
The following MDX member expressions are supported.

- <dimension>.CURRENTMEMBER
- <member>.FIRSTCHILD
- <member>.FIRSTSIBLING
- <member>.LAG
- <member>.LASTCHILD
- <member>.LASTSIBLING
- <member>.LEAD
- <member>.NEXTMEMBER
- <member>.PARENT
- <member>.PREVMEMBER

Supported Member Functions
The following MDX member functions are supported.

- ANCESTOR(...)
- COUSIN(...)

- OPENINGPERIOD(...)
- PARALLELPERIOD(...)

Supported Numeric Functions
The following MDX numeric functions are supported.

- AGGREGATE(...)
- AVG(...)
- CORRELATION(...)
- COUNT(...)
- COVARIANCE(...)
- LINREGINTERCEPT(...)
- LINREGPOINT(...)
- LINREGR2(...)
- LINREGSLOPE(...)
- LINREGVARIANCE(...)
- MAX(...)
- MEDIAN(...)
- MIN(...)
- RANK(...)
- STDDEV(...)
- SUM(...)
- VAR(...)

Supported Set Expressions
The following MDX set expressions are supported.

- <dimension>.MEMBERS
- <level>.MEMBERS
- <member>. CHILDREN
- <member>.SIBLINGS

Supported Set Functions
The following MDX set functions are supported.

- ADDCALCULATEDMEMBERS(...)
- BOTTOMCOUNT(...)
- BOTTOMPERCENT(...)
- BOTTOMSUM(...)
- CROSSJOIN(...)
- DESCENDANTS(...)
- DISTINCT(...)
- DRILLDOWNLEVEL(...)
- DRILLDOWNLEVELBOTTOM(...)
- DRILLDOWNLEVELTOP(...)
- DRILLDOWNMEMBER(...)
- DRILLDOWNMEMBERBOTTOM(...)
- DRILLDOWNMEMBERTOP(...)
- DRILLUPMEMBER(...)

- DRILLUPLEVEL(...)
- EXCEPT(...)
- EXTRACT(...)
- FILTER(...)
- GENERATE(...)
- HEAD(...)
- HIERARCHIZE(...)
- INTERSECT(...)
- LASTPERIODS(...)
- ORDER(...)
- PERIODSTODATE(...)
- TOPCOUNT(...)
- TOGGLEDRILLSTATE(...)
- TOPPERCENT(...)
- TOPSUM(...)
- SUBSET(...)
- UNION(...)

Supported Tuple Expressions
The following MDX tuple expressions are supported.

- <set>.CURRENTMEMBER
- <set>[.ITEM](...)

TM1 specific MDX functions

TM1 supports several TM1 specific MDX expressions. You can apply these expressions while developing MDX applications to run against the server or when creating/editing dynamic subsets in the Expression Window of the Subset Editor.

TM1FILTERBYPATTERN(<set>, <pattern_str>)

This TM1 specific MDX function returns all the members in <*set*> with names matching the pattern <*pattern_str*>.

Syntax

The syntax of <*pattern_str*> is the same used for the Select By Regular Expression option on the Subset Editor.

TM1FILTERBYLEVEL(<set>, <level_number>)

This TM1 specific MDX function returns all the members in <*set*> of the specified <*level_number*>.

Syntax

<*level_number*> is a number specifying the TM1 level number not an MDX level number.

TM1DRILLDOWNMEMBER(<set1>, <set2>|ALL [,RECURSIVE])

This TM1 specific MDX function is similar to the DRILLDOWNMEMBER function from Microsoft , but it has been adjusted to match the functionality of the Expand button {bmct expand_button.bmp} on the Subset Editor.

Syntax

ALL means drilldown all the members in <*set1*>.

RECURSIVE means that when one member from <*set1*> is being drilled down upon, every consolidated member resulting from that expansion will also be recursively drilled down until level 0 (TM1 level 0) is reached.

TM1Member
This function returns a member from a specified tuple.

A null member reference is returned when any of the following conditions are encountered:

- A null Tuple parameter
- An out-of-range numeric Index parameter
- A dimension or hierarchy parameter not found in the passed tuple.

Syntax

```
TM1Member(Tuple, MemberSpecifier);
```

Argument	Description
Tuple	An expression that resolves to a tuple.
MemberSpecifier	This parameter can be either a 0-based numeric index into the tuple or the name of a dimension/ hierarchy associated with the tuple. See below for examples showing both parameter types.

Example

```
TM1Member ( [model].Members.Item(23) ,0 ) ]
```

This example uses a numeric index into the tuple as the MemberSpecifier argument.

```
TM1Member( [model].Members.Item(23), [Model] ) ]
```

This example uses the name of a dimension associated with the tuple as the MemberSpecifier argument.

TM1SORT(<set>, ASC|DESC)
This TM1 specific MDX function sorts <*set*> alphabetically.

Syntax

ASC sorts A-Z

DESC sorts Z-A

TM1SORTBYINDEX(<set>, ASC|DESC)
This TM1 specific MDX function sorts <*set*> by the index value of the members.

Syntax

ASC sorts by ascending index value.

DESC sorts by descending index value.

TM1SUBSETALL([<dimname>])
This TM1 specific MDX function returns the TM1 subset All of <*dimname*>.

Syntax

```
TM1SubsetAll([<dimname>]);
```

TM1SubsetToSet

This function returns the members of a TM1 subset. TM1SubsetToSet is equivalent to the *<dimension>.<subsetname>* expression, but does not require string literals. Instead, TM1SubsetToSet lets you use expressions that resolve to the appropriate dimension and subset.

If a private and a public subset have identical names, enter the scope parameter to specify the scope of the search as "`private`" or "`public`".

Syntax

```
TM1SubsetToSet(Dimension_exp, Subset_exp, Scope);
```

Argument	Description
Dimension_exp	An expression that resolves to a valid TM1 dimension name. Use [dimension] or [dimension].[hierarchy] to specify a dimension.
Subset_exp	An expression that resolves to a valid subset of the dimension returned by Dimension_exp. When resolving an expression for a subset, the server searches first in the private subset list and then in the public list.
Scope (optional)	The subset list to search, which is specified as `private` or `public`. Specify `public` to search the public list only. If this parameter is not specified and a private and a public subset have the same name, the private subset is returned.

Example

MDX sample code with dimension name `[Corp Planning Hry]` in the last line:

```
{INTERSECT(EXCEPT(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),
TM1FILTERBYLEVEL(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),0)),{
TM1SubsetToSet([Corp Planning Hry],"elist")}),[Corp Planning Hry].[FixedAssets]}
```

MDX sample code with fully qualified dimension name `[Corp Planning Hry].[Corp Planning Hry]` and `[Corp Planning Hry].[FixedAssets]` in the last line:

```
{INTERSECT(EXCEPT(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),
TM1FILTERBYLEVEL(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),0)),{
TM1SubsetToSet([Corp Planning Hry].[Corp Planning Hry],"elist")}),
[Corp Planning Hry].[FixedAssets]}
```

Example

MDX sample code with two subsets, one private, one public, both named 'MySub' under a `Markets` dimension. The following queries on columns from [QB] return the content of the private subset:

```
select TM1SubsetToSet( [QBMarkets].[QBMarkets], 'MySub' )
```

Or:

```
select TM1SubsetToSet( [QBMarkets].[QBMarkets], 'MySub', 'private' )
```

The following query on columns from [QB] returns the content of the public subset:

```
select TM1SubsetToSet( [QBMarkets].[QBMarkets], 'MySub', 'public' )
```

TM1TupleSize

This function returns the number of members in a tuple.

Syntax

```
TM1TupleSize(Tuple);
```

Argument	Description
Tuple	An expression that resolves to a tuple. The function returns 0 if the Tuple argument does not resolve to a valid tuple, or of the tuple is null or empty.

TM1 specific MDX expressions

TM1 supports several TM1 specific MDX expressions. You can apply these expressions while developing MDX applications to run against the server or when creating/editing dynamic subsets in the Expression Window of the Subset Editor.

<dimension>.<subsetname>

This TM1 specific MDX expression returns members of <subsetname> in <dimension>.

Syntax

Since the same syntax (<dimension>.IDENTIFIER) is used for members and levels, a subset with the same name of a member or a level will never be instantiated.

When searching for a subset, the server searches first in the private subset list and then in the public list.

<member>.ANCESTORS

This TM1 specific MDX expression returns the ancestors of <member>.

Syntax

For example, assuming the following hierarchy of the Month dimension:

- Year
  - 1 Quarter
  - Jan
  - Feb
  - Mar

The following expression:

```
month.jan.ANCESTORS
```

returns the set

```
{ 1Quarter, Year }.
```

If the member has more than one immediate parent, the expression returns the set containing the first parent in the default hierarchy. Consider a hierarchy of a Region dimension, where the member Belgium has more than one immediate parent, being Benelux and Europe.

The following expression:

```
region.belgium.ANCESTORS
```

returns the set

```
{ Benelux, Europe }.
```

<member>.WEIGHT

This TM1 specific MDX expression returns the weight of <member>.

Syntax

The weighting property controls how a child member rolls up to its immediate parent, whether that child is also a parent of another consolidation or just a leaf member. Top level consolidations do not have weight.

Data spread keyboard shortcuts

The following tables include details for all spreading methods that you can use in Planning Analytics Workspace.

You can specify the direction for data spreading by using these codes:

Down

|

Up

^

Left

<

Right

>

The default data action is Replace. To specify subtract, use a tilde (~). Use a plus sign (+) to specify the Add data action.

The following list summarizes the spreading methods that you can use:

Proportional

P

Example:

P<>100

Proportionally spreads the value 100 to all leaf cells on the row of insertion, and replaces the existing cell values.

Equal spread

S

Example:

S+|^200

Equally spreads the value 200 to all leaf cells on the column of insertion, and adds the product of spreading to the existing cell values.

Repeat

R

Example:

R~<50

Subtracts the value 50 from all leaf cells to the left of the insertion point.

Clear

C

Example:

C|^<>

Clears values from all leaf cells in the view.

Percent change

P%

Example:

P%+|^<>10

Applies a percent change of 10% to all leaf values, adds the product to the existing cell values, and increments all leaves by 10%.

Repeat leaves

LR

Example:

LR+*2100

Adds 2100 to all leaves of the consolidation. If you omit *, it copies 2100 only to populated leaf cells.

Equal leaves

LS

Example:

LS500

Distributes 500 equally across populated leaves of a consolidation.

Straight line

SL

Example:

SL>100:200

Replaces all leaf values to the right of the point of insertion, with a start value of 100 and an end value of 200.

Can be used across a single row or column, but not across rectangular ranges.

Growth percentage

GR

Example:

GR|300:25

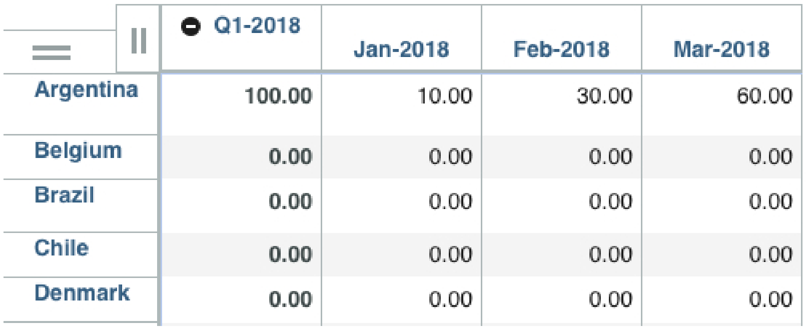
Applies a 25% growth percentage to the starting value of 300 and replaces all leaf values below the point of insertion.

Can be used across a single row or column, but not across rectangular ranges.

Proportional data spreading method

The Proportional spread method distributes a specified value among cells proportional to existing cell values.

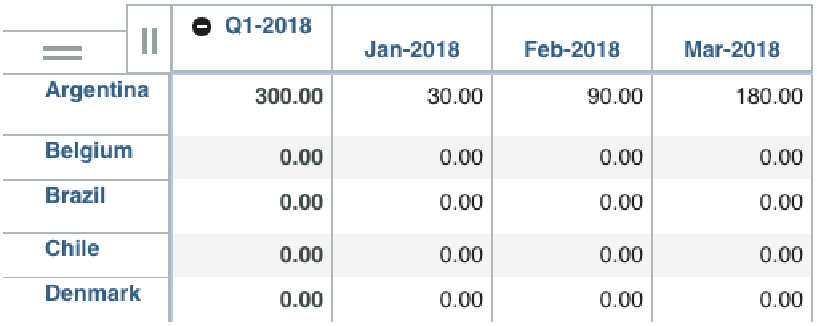
For example, consider the following view in which the values for Argentina in the months Jan, Feb, and Mar are 10, 30, and 60, respectively.



	⊖ Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	100.00	10.00	30.00	60.00
Belgium	0.00	0.00	0.00	0.00
Brazil	0.00	0.00	0.00	0.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

The sum of these values is 100, with the value in Jan accounting for 10% of the sum, the value in Feb accounting for 30%, and the value in Mar accounting for 60%.

When you proportionally distribute the value 300 across these cells and select the Replace update action, the result is as follows.



	⊖ Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	300.00	30.00	90.00	180.00
Belgium	0.00	0.00	0.00	0.00
Brazil	0.00	0.00	0.00	0.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

These values are proportionally equivalent to the values that existed before you apply data spreading.

- Jan contains the value 30, which is 10% of 300
- Feb contains the value 90, which is 30% of 300
- Mar contains the value 180, which is 60% of 300

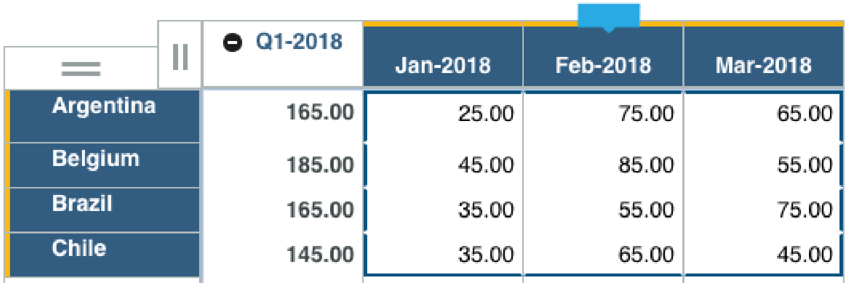
To distribute the value 300 proportionally across cells, equivalent to the values that existed before, type the following code:

```
P>300
```

Equal data spreading method

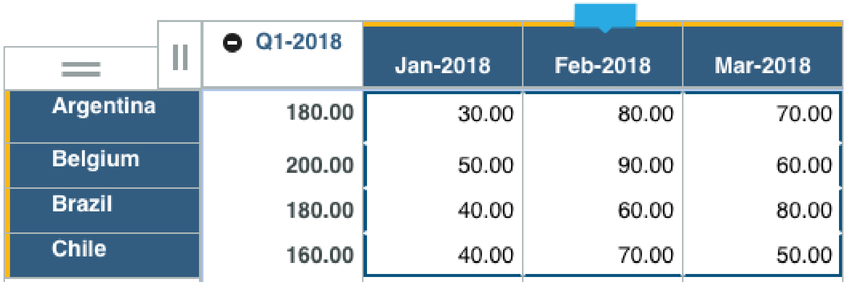
The Equal Spread method distributes a specified value equally across the cells in a view.

For example, consider the following view where a range of 12 cells is selected.



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	165.00	25.00	75.00	65.00
Belgium	185.00	45.00	85.00	55.00
Brazil	165.00	35.00	55.00	75.00
Chile	145.00	35.00	65.00	45.00

When you equally spread the value 60 to these cells and select the Add update action, the value is equally spread across the range and added to the existing cell values. The result is that each cell value is incremented by 5 (60/12=5).



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	180.00	30.00	80.00	70.00
Belgium	200.00	50.00	90.00	60.00
Brazil	180.00	40.00	60.00	80.00
Chile	160.00	40.00	70.00	50.00

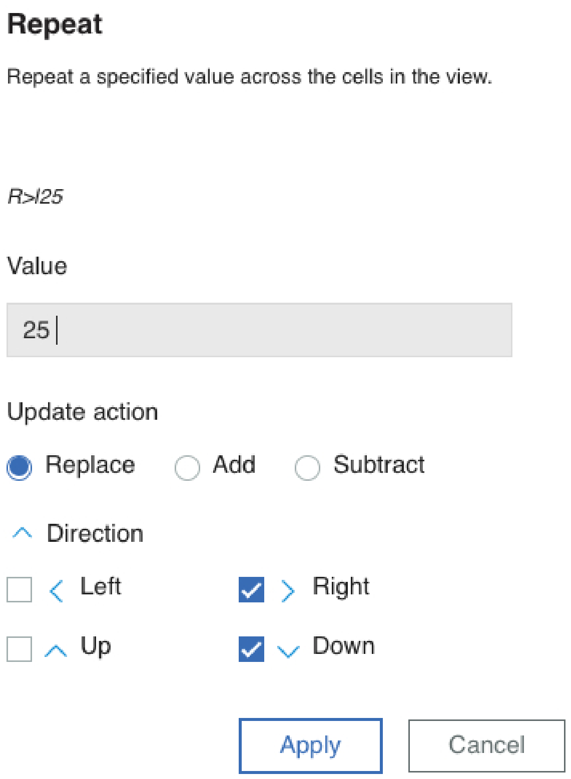
To spread the value 60 equally across cells to the right and down, type the following code:

```
S>|60
```

Repeat data spreading method

The Repeat method repeats a specified value across cells in a view.

For example, right-click a cell at the intersection of Brazil and February, and select Repeat for the value 25 in the Right in the Right and Down directions:



Repeat

Repeat a specified value across the cells in the view.

R>|25

Value

25

Update action

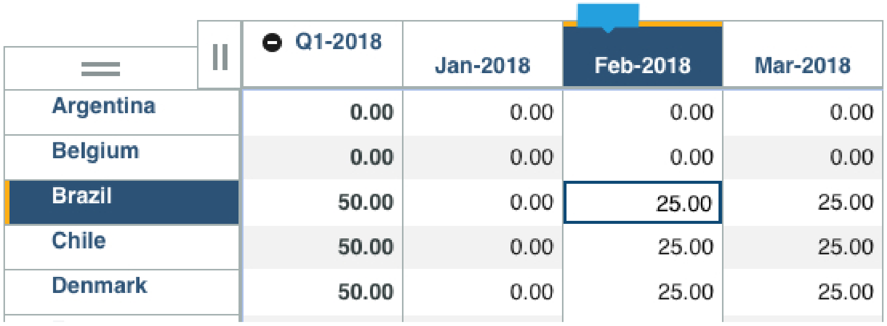
● Replace ○ Add ○ Subtract

Direction

□ Left ☑ Right

□ Up ☑ Down

This results in the following view:



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	0.00	0.00	0.00	0.00
Belgium	0.00	0.00	0.00	0.00
Brazil	50.00	0.00	25.00	25.00
Chile	50.00	0.00	25.00	25.00
Denmark	50.00	0.00	25.00	25.00

Clear data spreading method

The Clear method clears values from cells in a view. You can apply this method to either leaf cells or consolidated cells. When you apply the Clear method to a consolidated cell, all leaves of the consolidation get set to zero.

If you initiate the Clear method from the selected cell and extend the spreading operation downward, the leaves of all consolidations below the point of insertion are cleared.

Data entry shortcut code

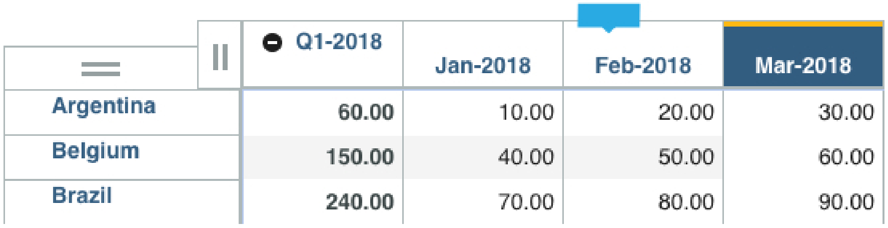
To clear values to the left and down, type this code:

```
c<|
```

Percent change data spreading method

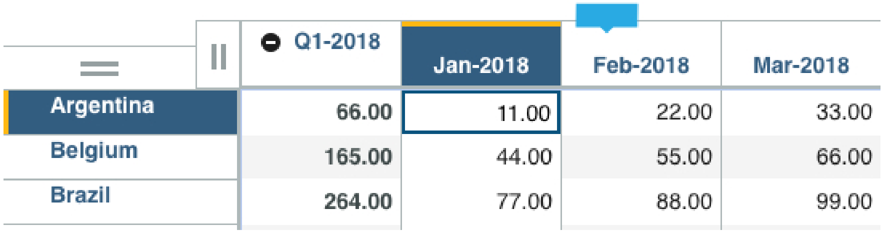
The Percent Change method multiplies the current cell values by a specified percentage. The product of that multiplication can then replace, be added to, or be subtracted from the existing cell values.

For example, consider the following view that contains a range of values in increments of 10.



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	60.00	10.00	20.00	30.00
Belgium	150.00	40.00	50.00	60.00
Brazil	240.00	70.00	80.00	90.00

When you apply the Percent Change method to these cells and specify a % Change value of 10, the system multiplies each cell value by 10% (or .10). If you select the Add update action, the product of multiplication is added to the existing cell values. The result is that each cell value is incremented by 10%.



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	66.00	11.00	22.00	33.00
Belgium	165.00	44.00	55.00	66.00
Brazil	264.00	77.00	88.00	99.00

Data entry shortcut code

To increase the value of each cell by 10% to the right and down, type this code:

```
P%+>|10
```

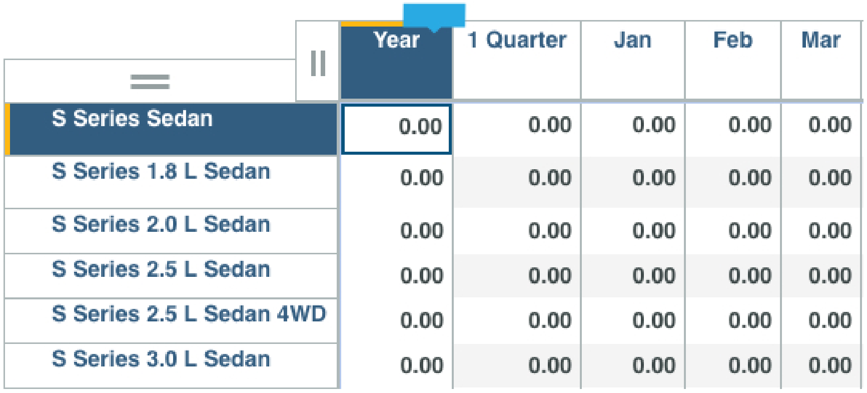
Repeat leaves data spreading method

The Repeat Leaves method copies a specified value to the leaves (children) of a consolidation. When you apply this method, you can copy the value to all leaves of the consolidation or only to those leaves that already contain non-zero values.

For example, say there are several leaves of Year, Argentina with values.

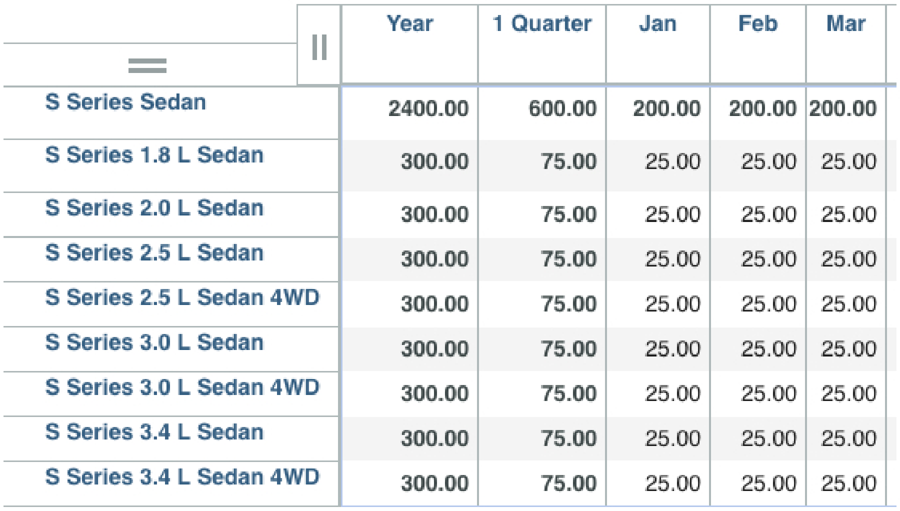
If you use the Repeat Leaves method to copy the value 400 to the leaves of Year, Argentina currently populated with non-zero values, the value 400 is copied to all leaves that contained non-zero values.

If you initiate the Repeat Leaves method from a cell identified by more than one consolidated member, the specified value is copied to all leaves associated with the cell. For example, in the following view, the selected cell is identified by two consolidated members: Year and S Series Sedan.



	Year	1 Quarter	Jan	Feb	Mar
S Series Sedan	0.00	0.00	0.00	0.00	0.00
S Series 1.8 L Sedan	0.00	0.00	0.00	0.00	0.00
S Series 2.0 L Sedan	0.00	0.00	0.00	0.00	0.00
S Series 2.5 L Sedan	0.00	0.00	0.00	0.00	0.00
S Series 2.5 L Sedan 4WD	0.00	0.00	0.00	0.00	0.00
S Series 3.0 L Sedan	0.00	0.00	0.00	0.00	0.00

If you initiate Repeat Leaves spreading from the highlighted cell, the specified value is copied to all cells identified by the leaves of Year and the leaves of S Series Sedan. For instance, if you use Repeat Leaves to copy the value 25 to all leaves of the highlighted cell, the result is as follows.



	Year	1 Quarter	Jan	Feb	Mar
S Series Sedan	2400.00	600.00	200.00	200.00	200.00
S Series 1.8 L Sedan	300.00	75.00	25.00	25.00	25.00
S Series 2.0 L Sedan	300.00	75.00	25.00	25.00	25.00
S Series 2.5 L Sedan	300.00	75.00	25.00	25.00	25.00
S Series 2.5 L Sedan 4WD	300.00	75.00	25.00	25.00	25.00
S Series 3.0 L Sedan	300.00	75.00	25.00	25.00	25.00
S Series 3.0 L Sedan 4WD	300.00	75.00	25.00	25.00	25.00
S Series 3.4 L Sedan	300.00	75.00	25.00	25.00	25.00
S Series 3.4 L Sedan 4WD	300.00	75.00	25.00	25.00	25.00

Data entry shortcut code

To copy the value of 25 to each leaf of a consolidation, whether populated or not, type the following code

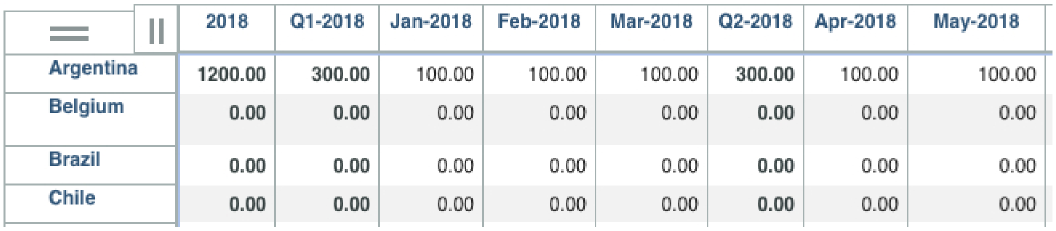
```
LR*25
```

Equal leaves data spreading method

The Equal leaves spreading method distributes a specified value equally across all leaves of a consolidated cell. When you apply this method, you can choose to distribute the value to all leaves of the consolidation or only to those leaves that already contain non-zero values.

In this example, there are several leaves of Year, Argentina with zero values.

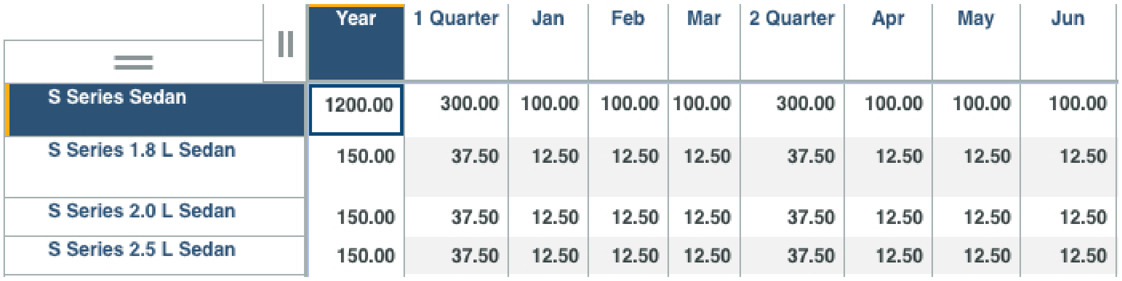
If you use the equal leaves method to distribute the value 1200 to all the leaves of Year, Argentina, the result is as follows.



	2018	Q1-2018	Jan-2018	Feb-2018	Mar-2018	Q2-2018	Apr-2018	May-2018
Argentina	1200.00	300.00	100.00	100.00	100.00	300.00	100.00	100.00
Belgium	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Brazil	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Chile	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

If you initiate the Equal leaves spreading method from a cell identified by more than one consolidated member, the value is distributed to all leaves associated with the cell. For example, say the selected cell is identified by two consolidated members: Year and S Series Sedan.

If you initiate Equal leaves spreading from the highlighted cell, the specified value is distributed to all cells identified by the leaves of Year and the leaves of S Series Sedan. For instance, if you use Equal leaves spreading to distribute the value 1200 to all leaves of the highlighted cell, the result is as follows.



	Year	1 Quarter	Jan	Feb	Mar	2 Quarter	Apr	May	Jun
S Series Sedan	1200.00	300.00	100.00	100.00	100.00	300.00	100.00	100.00	100.00
S Series 1.8 L Sedan	150.00	37.50	12.50	12.50	12.50	37.50	12.50	12.50	12.50
S Series 2.0 L Sedan	150.00	37.50	12.50	12.50	12.50	37.50	12.50	12.50	12.50
S Series 2.5 L Sedan	150.00	37.50	12.50	12.50	12.50	37.50	12.50	12.50	12.50

When you initiate Equal leaves spreading from a cell identified by multiple consolidated members, the RAM requirements of the cube can increase significantly. Where more than 10,000 cells are affected by the Equal leaves method, a warning is displayed. In circumstances where more than one million cells are affected, the spreading operation does not execute.

Data entry shortcut code

To distribute the value of 1200 equally across all leaves of a consolidation, type this code:

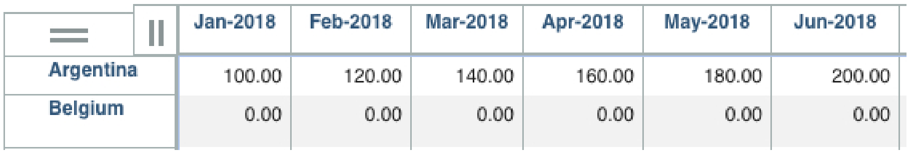
```
LS*1200
```

Straight line data spreading method

The Straight line data spreading method populates cube cells by linear interpolation between two specified endpoints.

For example: if you have a start value of 100 and an end value of 200, the midpoint value would be 150.

The following view shows the effect of Straight line spreading across a range of six cells with the start value of 100 and and an end value of 200.



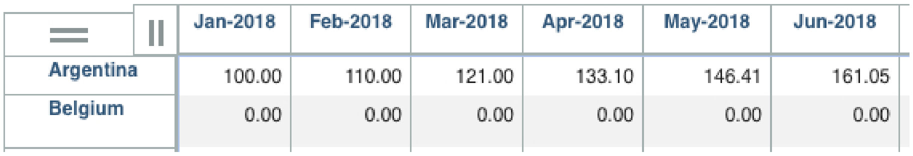
	Jan-2018	Feb-2018	Mar-2018	Apr-2018	May-2018	Jun-2018
Argentina	100.00	120.00	140.00	160.00	180.00	200.00
Belgium	0.00	0.00	0.00	0.00	0.00	0.00

With the start value of 100 and the end value of 200, the option populates the cells between the start and the end cells with values at equal intervals.

Growth percent data spreading method

The Growth percent method takes an initial value as a starting point and sequentially increments all values in a selected range of cells by the specified percentage.

The following view shows the result of applying the Growth percent method to a range of six cells where the initial value is 100 and the growth percentage is 10%. This example uses the Replace data action.



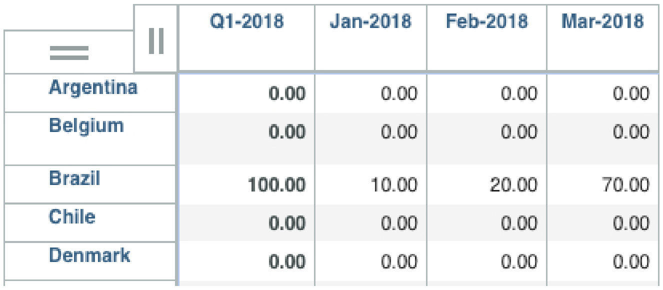
	Jan-2018	Feb-2018	Mar-2018	Apr-2018	May-2018	Jun-2018
Argentina	100.00	110.00	121.00	133.10	146.41	161.05
Belgium	0.00	0.00	0.00	0.00	0.00	0.00

The initial value of 100 displays in the cell identified by Jan-2018, Argentina. Applying the growth percentage of 10% to 100 gives 110, the value in Feb-2018, Argentina. Applying the growth percentage of 10% to 110 yields 121, the value in Mar-2018, Argentina.

Relative proportional data spreading method

Use the Relative Proportional Spread method to spread values to the leaves of a consolidation cell proportional to the leaves of a reference cell.

The following example demonstrates relative proportional spreading where both the initial cell and the reference cell are in the same cube.



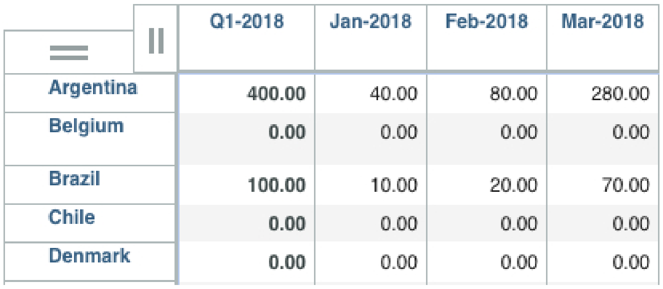
	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	0.00	0.00	0.00	0.00
Belgium	0.00	0.00	0.00	0.00
Brazil	100.00	10.00	20.00	70.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

The reference cell can be located in the cube from which you initiate spreading or in a separate cube. However, the reference cell must share the same consolidations as the cell from which you initiate spreading.

The above view shows a single consolidated value of 100 in the cell identified by 1 Quarter, Brazil. By looking at the leaves of 1 Quarter, you can see that:

- Jan contains the value 10, which is 10% of 100
- Feb contains the value 20, which is 20% of 100
- Mar contains the value 70, which is 70% of 100

If you initiate relative proportional spreading from 1 Quarter, Argentina and specify a Data Action of Replace when spreading the value 400, the leaves of 1 Quarter, Argentina are populated as proportional to the leaves of 1 Quarter, Brazil:



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	400.00	40.00	80.00	280.00
Belgium	0.00	0.00	0.00	0.00
Brazil	100.00	10.00	20.00	70.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

- Jan contains the value 40, which is 10% of 400
- Feb contains the value 80, which is 20% of 400
- Mar contains the value 280, which is 70% of 400

Relative percent adjustment data spreading method

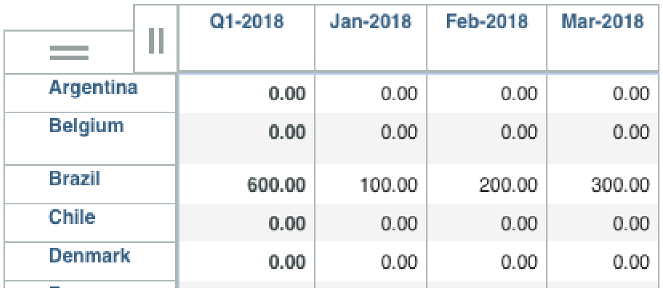
The Relative Percent Adjustment method spreads values to the leaves of a consolidation by applying a percentage adjustment to the leaves of a reference cell.

This method increments the values in the leaves of the reference cell by a specified percentage. The resulting values are then spread to the leaves of the consolidation from which you initiated spreading.

The reference cell can be located in the cube from which you initiate spreading or in a separate cube. However, the reference cell must share the same consolidations as the cell from which you initiate spreading.

The following example illustrates relative percent adjustment spreading where both the initial cell and the reference cell exist in the same cube.

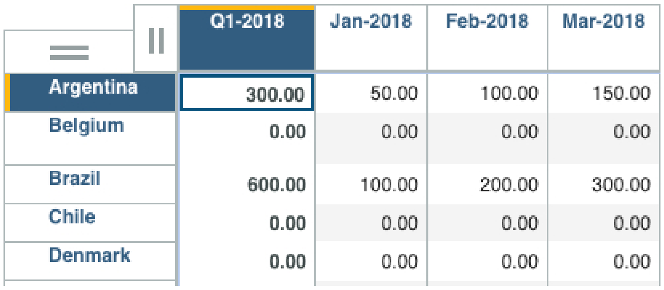
Say you have a single consolidated value of 600 is in the cell identified by 1 Quarter, Brazil. The leaves of 1 Quarter would look like this:



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	0.00	0.00	0.00	0.00
Belgium	0.00	0.00	0.00	0.00
Brazil	600.00	100.00	200.00	300.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

- Jan contains the value 100
- Feb contains the value 200
- Mar contains the value 300

If you initiate Relative Percent Adjustment spreading from 1 Quarter, Argentina and specify a percent adjustment of 50% while using 1 Quarter, Brazil as a reference cell, the result is as follows.



	Q1-2018	Jan-2018	Feb-2018	Mar-2018
Argentina	300.00	50.00	100.00	150.00
Belgium	0.00	0.00	0.00	0.00
Brazil	600.00	100.00	200.00	300.00
Chile	0.00	0.00	0.00	0.00
Denmark	0.00	0.00	0.00	0.00

Note that the leaves of 1 Quarter, Argentina are populated with values that are 50% of the values found in the leaves of 1 Quarter, Brazil:

- Jan, Argentina contains the value 50, a 50% adjustment of the value in Jan, Brazil
- Feb, Argentina contains the value 100, a 50% adjustment of the value in Feb, Brazil
- Mar, Argentina contains the value 150, a 50% adjustment of the value in Mar, Brazil

Exploration navigation shortcuts

You can use keystrokes to move to different locations in an exploration (table) view.

Table 11. Exploration navigation shortcuts

Keystroke	**Navigation result**
Up, **down**, **left**, or **right arrow** keys	Moves to the adjacent cell in the direction represented by the arrow key.
Backspace	Clears the content of the active cell. In cell editing mode, Backspace deletes the character to the left of the insertion point or deletes selected text.
Delete	Clears the content of the active cell. In cell editing mode, Delete deletes the character to the right of the insertion point or deletes selected text.
Esc	When in cell editing mode, Esc cancels the editing action.
Ctrl + Home	Moves to the first cell of the exploration.
Ctrl + End	Moves to the last cell of the exploration.
Page Up	Moves one screen up in the exploration.

Table 11. Exploration navigation shortcuts (continued)

Keystroke	Navigation result
Alt + Page Up	Moves one screen to the left in the exploration.
Page Down	Moves one screen down in the exploration.
Alt + Page Down	Moves one screen to the right in the exploration.
F2	Opens the current cell in editing mode.
Ctrl + right arrow key	Moves to the last visible cell in the current row.
Ctrl + left arrow key	Moves to the first visible cell in the current row.
Ctrl + down arrow key	Moves to the last visible cell in the current column.
Ctrl + up arrow key	Moves to the first visible cell in the current column.

Naming conventions

IBM Planning Analytics has the following naming conventions. Some of these conventions are enforced.

Although some of these characters are not reserved, it is a good practice to avoid the use of these special characters in most cases when you name objects and members. For more information, see “Member names and MDX expressions” on page 645.

Table 12. Special characters to avoid in object and member names

Character	Description
’	apostrophe
*	asterisk
@	at sign - see “Object names in TM1 rules” on page 646.
\	back-slash
:	colon
,	comma
{	curly brace - see “The curly brace in object names” on page 645.
"	double-quote
!	exclamation mark - see “Object names in TM1 rules” on page 646.
>	greater-than
<	less-than
-	minus sign - in member names. See “Member names and MDX expressions” on page 645.
| or ¦	pipe or broken pipe

Table 12. Special characters to avoid in object and member names (continued)

Character	Description
+	plus sign - in member names. See “Member names and MDX expressions” on page 645.
?	question-mark
;	semicolon
/	slash
~	tilde - see “Object names in TM1 rules” on page 646.
^	caret - see “Member names and MDX expressions” on page 645.

Reserved characters per component

The following characters are explicitly reserved for the following components and must never be used when you name objects in these contexts:

- TM1 Architect reserves the following characters:

```
\ / : * ? " < > | }
```

- TM1 Server reserves these characters in these objects: Cube, Dimension, Subset, View, Process, Chores.

```
\ / : * ? " < > | ' ; ,
```

- For process variable names, the identifier cannot contain any special characters except for:

```
AllowableChars[] = ".$%_`";
```

The curly brace in object names

It is a good practice to avoid the use of the right curly brace (}) as the first character in any user-created TM1 object name. TM1 control object names always begin with the right curly brace. If a user-created object name begins with a right curly brace, the object becomes hidden if the **Display Control Objects** parameter is turned off.

Member names and MDX expressions

Do not use + or - as the first character of a member name. Although only the first member in a set when slicing to active form cannot use + or - as the first character in the element name, it is a good practice to never use + or - as the first character of an member name.

Do not use a ^ in a member name. The ^ character can be used as the delimiter between the ancestor and multi-parent member's name but when a member name that contains this character is referenced in an MDX expression, it cannot be escaped.

Although all the other characters available for use in member names are technically not restricted, it is good practice to avoid the special characters that are listed in the previous table when you name members.

A member name can contain a right square bracket ( ]) but when a member name that contains this character is referenced in an MDX expression, the character needs to be escaped by doubling it. For example, a member that is named Array[N] Elements, can be referred to in an MDX expression as [Array[N]] Elements].

Object names in TM1 rules

Although technically allowed, it is a good practice to avoid using these special characters in object names because they may conflict when used in a rules expression. This guideline protects you if the objects or members ever become part of a rule statement where those special characters are not permitted.

- For example,) | ~ ; @ \ / : * ? " < > are all often found in rules statements and should not be used in object names.
- The @ is technically not restricted, however it is a good practice to avoid using the @ character in object names or member names because the @ character is also a string comparison operator in TM1 rules. If you reference any object with a name that contains the @ character in rules, the object name must be enclosed in single quotation marks. For example, a dimension named products@location must be referenced as 'products@location' in rules. Escaping the name with quotation marks does not work in all cases, so it is best to avoid the use of @ in all cases when naming objects.
- Escaping the special character using quotation marks does not work for ! or in certain rule expressions.
- The exclamation point ! character must not be used in object names because it is also used in rules expressions. For example:

  ```
  DB('MarketExchange',!market,!date)
  ```

Maximum string length for data directory and object names

The entire string that is represented by the combination of the TM1 Server data directory name and the object name is limited to 128 bytes. For example, if your data directory is C:\Financial data\TM1\ (22 bytes), object names are limited to 106 bytes, inclusive of a file extension such as .cub or .rux.

Some TM1 objects, such as views, subsets, and applications, are stored in subdirectories of the TM1 Server data directory. In this case, the 128-byte limit is applied to the combination of the TM1 Server data directory, the subdirectory, and the object name.

Case sensitivity

Object names are not case-sensitive. For example, the dimension name actvsbud is equivalent to ActVsBud.

Spaces in object names

Spaces are allowed in all object names, but spaces are ignored by the TM1 Server. The TM1 Server considers the dimension name Act Vs Bud to be equivalent to ActVsBud (or actvsbud).

User names

User names that include reserved characters cannot save private objects.

Control objects

Control objects perform special tasks in IBM Planning Analytics. Control objects are generated by the system. Only modelers and administrators can see control objects.

Control objects include cubes, processes, and dimensions. Their names are prefixed with }, for example: }Capabilities. Control objects are available from the **Control Objects** node in the content tree.

You can view control cubes by right-clicking a control cube and selecting **Add new view**. You can't create control cubes because they are generated by the system for different purposes. To find out more about control cubes, see Control Cubes.

Control dimensions are used to track performance statistics, administer security, manage clients and groups, and store object attributes and properties. You can't edit control dimensions. To find out more about control dimensions, see Control Dimensions.

You can view control processes by right-clicking a process and selecting **Edit process**.

Map reference info

Planning Analytics Workspace supports specific geographic entity values and languages for maps.

Supported languages for maps

Maps in Planning Analytics Workspace support continent, county, and local state and province names in a number of different languages.

Planning Analytics Workspace supports all continent and country names in the following list of languages. Planning Analytics Workspace also support local state and province names in their respective local languages for the languages on this list. For example, the name of the state of North Carolina in the United States is supported only in English. It is not supported in French as "Caroline du Nord" or in any other language.

Supported languages:

1. Catalan
2. Chinese-simplified (China)
3. Chinese-traditional (Taiwan)
4. Croatian
5. Czech
6. Danish
7. Dutch
8. English
9. Finnish
10. French
11. German
12. Greek
13. Hungarian
14. Italian
15. Japanese
16. Kazakh
17. Korean
18. Norwegian Bokmål
19. Polish
20. Portuguese (Brazil)
21. Romanian
22. Russian
23. Slovak
24. Slovenian
25. Spanish
26. Swedish
27. Thai
28. Turkish

Troubleshooting Planning Analytics Workspace

You can use these topics to help you get the best experience in IBM Planning Analytics Workspace.

Setting the TM1 Web session timeout

The default TM1 Web session timeout is 20 minutes. When TM1 websheets are deployed to IBM Planning Analytics Workspace, you might encounter TM1 Web session timeouts. You can modify this setting in your environment.

About this task

When TM1 websheets are deployed to Planning Analytics Workspace, the recommended session timeout is 60 minutes.

Note: As of IBM Planning Analytics Local version 2.0.6, you must **not** change the `session-timeout` value in the `web.xml` file.

In IBM Planning Analytics Local version 2.0.6, there is a parameter in the `tm1web_config.xml` file called `HttpSessionTimeout`. You can use this parameter to customize the session timeout (in minutes) of the HTTP session for TM1 Web.

If the `HttpSessionTimeout` parameter is not specified (missing or blank), the value is less than 1 or not a numerical value, the default `session-timeout` that is defined in the `web.xml` file is used.

If you are using IBM Planning Analytics Local version 2.0.6 or later, to customize the session timeout for TM1 Web, set the `HttpSessionTimeout` parameter in `tm1web_config.xml`. See step "1" on page 648.

If you are using IBM Planning Analytics Local version 2.0.5 or earlier, to change the default session timeout, set the `<session-timeout>` parameter in `web.xml`. See step "2" on page 648.

Procedure

1. To customize the session timeout, follow these steps.
   a) Open the`tm1web_config.xml` in a text editor.

      The `tm1web_config.xml` file is located in your `<pa_installation_directory>\webapps\tm1web\WEB-INF\configuration` directory.

      For example, `C:\Program Files\IBM\cognos\tm1_64\webapps\tm1web\WEB-INF\configuration`.

   b) Change the `HttpSessionTimeout` to 60 or a value that is required by your environment.

      ```
      <add key="HttpSessionTimeout" value="60" />
      ```

   c) Save and close the `tm1web_config.xml`.
2. If you are using IBM Planning Analytics Local version 2.0.5 or earlier, to change the default session timeout, follow these steps.
   a) Open `web.xml` in a text editor.

      The `web.xml` file is located in your `<pa_installation_directory>\webapps\tm1web\WEB-INF` directory.

      For example, `C:\Program Files\IBM\cognos\tm1_64\webapps\tm1web\WEB-INF`.

   b) Change the `<session-timeout>` value to 60 or a value that is required by your environment.

      ```
      <session-config>
          <session-timeout>20</session-timeout>
      </session-config>
      ```

   c) Save and close `web.xml`.
3. Restart the IBM TM1 Application Server service.

Slow scrolling in Google Chrome browser

Scrolling in the Planning Analytics Workspace Content Tree or in cube views can be slow or unresponsive when using the default Chrome browser setting for Smooth Scrolling. To resolve this issue, modify the Smooth Scrolling setting from **Default** to **Disabled**.

Procedure

1. Type `chrome://flags` in the Chrome address bar.
2. Scroll down to find the **Smooth Scrolling** setting.
3. Change the setting from **Default** to **Disabled**.
4. Restart Chrome.

Help error: Unable to get resource file

When you view a topic in the IBM Planning Analytics Workspace integrated help system, you might see an error message:`Unable to get resource file - https://www.ibm.com/support/knowledgecenter/SSD29G_2.0.0/com.ibm.swg.ba.cognos.tm1_prism_gs.2.0.0.doc/`*`filename`*.

About this task

If this message appears, there may be a problem with the Knowledge Center.

Procedure

Try one of the following solutions:

- Wait a few minutes and then try selecting the topic again.
- Copy and paste the URL in the error message into your browser.

Accessibility

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

The major accessibility features are described in the following list:

- You can use the keyboard to work in the set editor. To find out more, see "Use the keyboard to work in the set editor" on page 150.
- You can use the keyboard to work in the conditional formatting editor. To find out more, see "Use the keyboard to work with conditional formatting" on page 133.
- You can use command keys, or shortcut keys, to navigate through IBM Planning Analytics Workspace. Shortcut keys directly trigger an action and usually use the CTRL or Cmd keys. For example, to save a workbook, use CTRL+S.
- Planning Analytics Workspace uses Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA). This means that people with limited vision can use screen-reader software, along with a digital speech synthesizer, to listen to what is displayed on the screen.
- You can use a high-contrast display in Planning Analytics Workspace.

For more information about the commitment that IBM has to accessibility, see IBM Accessibility on the web (www.ibm.com/able).

Notices

This information was developed for products and services offered worldwide.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing

3755 Riverside Dr.
Ottawa, ON
K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information here is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

©

Product Information

This document applies to IBM Planning Analytics version 2.0.0 and may also apply to subsequent releases.

Copyright

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web in " Copyright and trademark information " at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Microsoft product screen shot(s) used with permission from Microsoft.