IBM Planning Analytics
Version 11 Release 0

TM1 for Developers

IBM

Note

Before you use this information and the product it supports, read the information in Notices.

Product Information

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

Licensed Materials - Property of IBM

© Copyright International Business Machines Corporation 2007, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 6. Importing Data with Processing Worksheets........85

Chapter 7. Controlling Access to TM1 Objects........ 95

Chapter 8. Using TM1 Action Buttons to Build Worksheet Applications........ 107

Introduction

This documentation is intended for use with IBM® Cognos® TM1®.

This documentation describes how to create and maintain objects on the IBM Cognos TM1 server, administer security, and develop TM1 applications. The documentation also describes aspects of TM1 security.

Finding information

To find documentation on the web, including all translated documentation, access IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter).

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.

Accessibility features

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

This product does not currently support accessibility features that help users with a physical disability, such as restricted mobility or limited vision, to use this product.

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.

Chapter 1. Introduction to TM1 Development

This section provides an introduction to the concept of multi-dimensionality and describes some common responsibilities of developers that use IBM Cognos TM1.

Understanding Multi-dimensionality

With IBM Cognos TM1, you can create multidimensional databases that provide business and finance managers instant meaning from complex, dynamic business models.

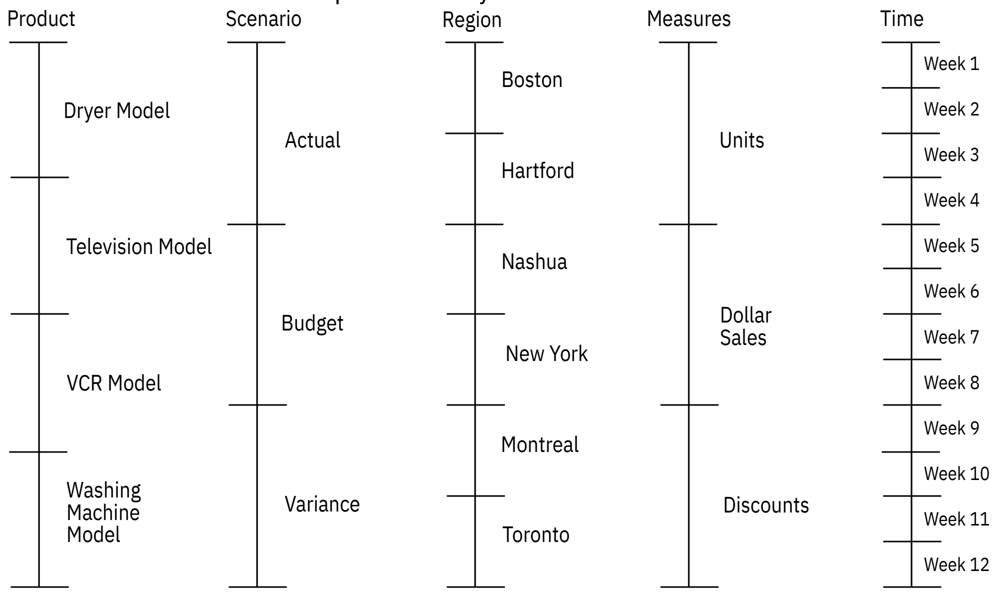
To understand multi-dimensionality, consider the example of the Vice President of Sales for a retail company who wants to analyze product sales across a retail chain that operates in the United States and Canada. Each retail store records the unit sales, dollar sales, and discounts for the durable consumer products.

The sales are analyzed by product, scenario (actual versus budget), region, measures (units, dollar sales, and discounts), and week. This example identifies a five-dimensional model. The dimensions identify how the data is organized or how the types of data are tracked.

In TM1, the sales analysis can reside in one or more multidimensional structures called cubes. A collection of cubes forms a database. Each data point in a cube is identified by one *element* in each dimension of the cube. For example, actual dollar sales of dryers during the second week of January in the Boston store. TM1 cubes must contain no less than two and no more than 256 dimensions.

Durables Cube

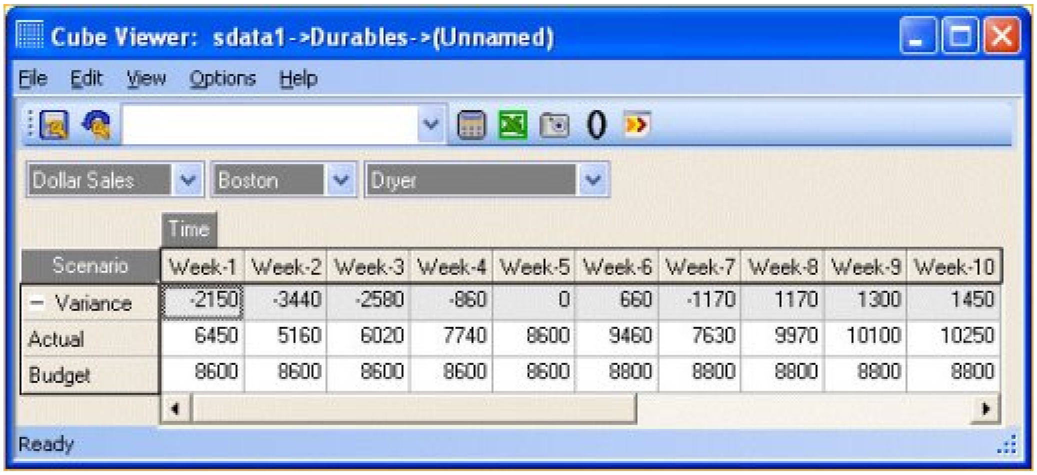
In the following diagram, each dimension in the Durables cube is represented by a vertical line segment. The elements within the dimension are represented by unit intervals.



Suppose that you are the Vice President of Sales, and you need to quickly compare the performance of products and stores to identify the winning strategies and trouble spots. Using TM1 multidimensional views, you can create an unlimited number of ad hoc queries.

In the following example, you can quickly compare actual versus budgeted dollar sales across weeks. The region is Boston and the product is a dryer model.

© Copyright IBM Corp. 2007, 2017

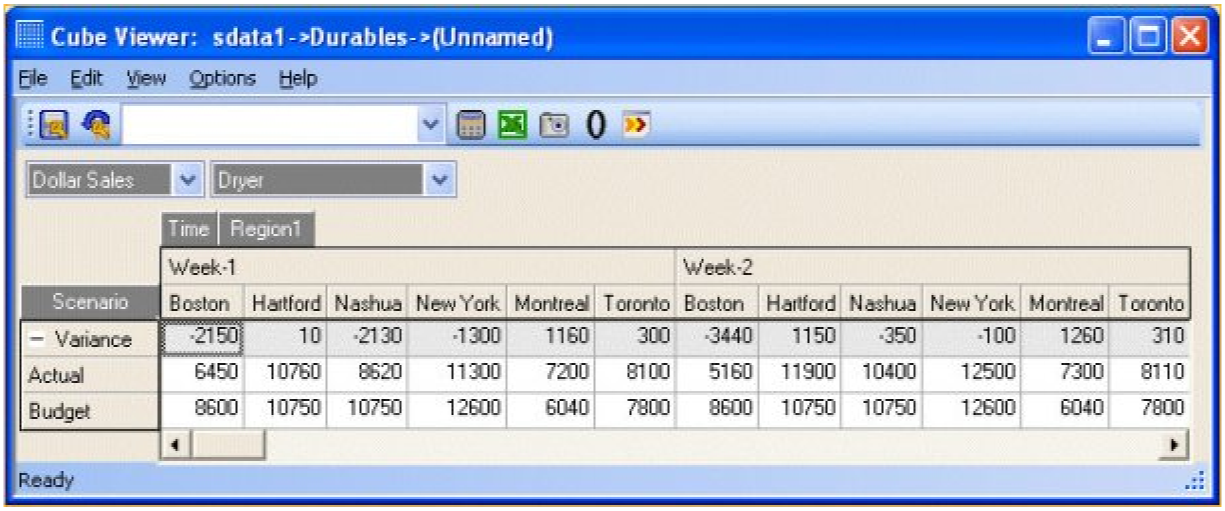


Dollar Sales | Boston | Dryer

Time

Scenario	Week-1	Week-2	Week-3	Week-4	Week-5	Week-6	Week-7	Week-8	Week-9	Week-10
− Variance	-2150	-3440	-2580	-860	0	660	-1170	1170	1300	1450
Actual	6450	5160	6020	7740	8600	9460	7630	9970	10100	10250
Budget	8600	8600	8600	8600	8600	8800	8800	8800	8800	8800

By rearranging the view, you can compare dollar sales for the dryer model across all regions.



Dollar Sales | Dryer

Time | Region1

	Week-1						Week-2					
Scenario	Boston	Hartford	Nashua	New York	Montreal	Toronto	Boston	Hartford	Nashua	New York	Montreal	Toronto
− Variance	-2150	10	-2130	-1300	1160	300	-3440	1150	-350	-100	1260	310
Actual	6450	10760	8620	11300	7200	8100	5160	11900	10400	12500	7300	8110
Budget	8600	10750	10750	12600	6040	7800	8600	10750	10750	12600	6040	7800

You can use TM1 to reconfigure views and drill down into your multi-dimensional data to satisfy your analysis requirements.

Your Role as Developer

As an IBM Cognos TM1 developer, your responsibilities fall into four major tasks.

- Design and create the cubes that hold business analysis.
- Decide where to store the cubes so they can be shared across the organization.
- Import data into the cubes from transactional systems and other data sources.
- Create formulas that do calculations, such as average prices, currency conversions, and price/earning ratios.

You must have access rights to the TM1 data to do these tasks. Typically, your TM1 administrator is responsible for setting access rights. The following section describes the distinction between a local and a remote server, and lists the access rights that you need to do tasks on a remote server.

TM1 object naming conventions

As a developer, you are responsible for creating and naming many objects in the IBM Cognos TM1 system. TM1 enforces some restrictions on naming while other guidelines offer best practices. Observe the following conventions when you name TM1 objects.

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 elements. See “Element names and MDX expressions” on page 4 for specific information about naming elements.

Table 1: Special characters to avoid in object and element names

Character	**Description**
’	apostrophe
*	asterisk
@	at sign - see “Object names in TM1 rules” on page 4.
\	back-slash
:	colon
,	comma
{	curly brace - see “The curly brace in object names” on page 4.
"	double-quote
!	exclamation mark - see “Object names in TM1 rules” on page 4.
>	greater-than
<	less-than
-	minus sign - in element names. See “Element names and MDX expressions” on page 4.
|	pipe
+	plus sign - in element names. See “Element names and MDX expressions” on page 4.
?	question-mark
;	semicolon
/	slash
~	tilde - see “Object names in TM1 rules” on page 4.

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 name, 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.

Element names and MDX expressions

Do not use + or - as the first character of an element name. Although only the first element in a subset 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 element name.

Although all the other characters available for use in element names are technically not restricted, it is good practice to avoid the special characters that are listed in the previous table when you name elements.

An element name can contain a right square bracket ( ]) but when an element name that contains this character is referenced in an MDX expression, the character needs to be escaped by doubling it. For example, an element 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 elements 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 element 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 IBM Cognos 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 IBM Cognos 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.

Chapter 2. Creating Cubes and Dimensions

IBM Cognos TM1 stores business data in cubes. This documentation describes how to create cubes and their building blocks, dimensions.

Note: All tasks that are described in this documentation need TM1 Perspectives or TM1 Architect. You cannot create cubes, create or edit dimensions, or establish replications with TM1 Client.

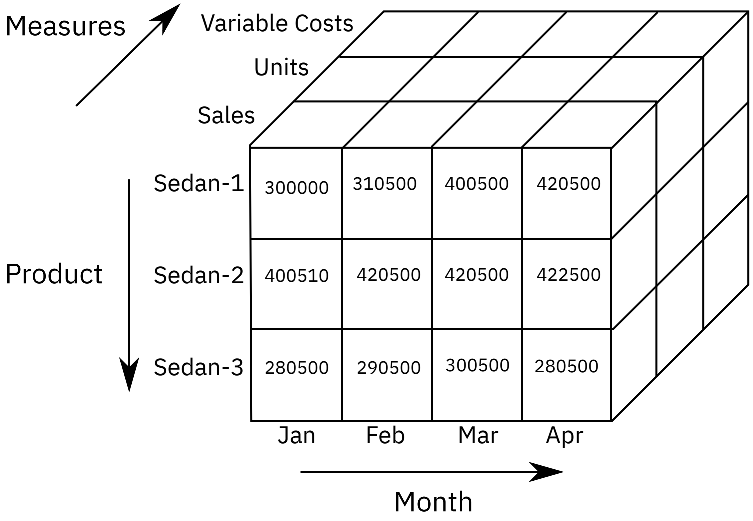
Designing Cubes

TM1 stores your business analysis in cubes. Each cell in a cube contains a measure that you are tracking in an analysis. A cube can store data against one or more measures.

You form a cube with dimensions, which identify how to organize the data or the measures you want to track. One element in each dimension identifies the location of a cell in a cube.

The following example cube contains three dimensions: Product, Measures, and Month. Each measure, such as Sales, is organized or dimensioned by a product and a month. For example, the cell value 300000 represents the sales of Sedan-1 in the month of January (Jan).

TM1 treats all dimensions the same way, whether they contain elements that identify measures or describe how you organize the measures.



	Jan	Feb	Mar	Apr
Sedan-1	300000	310500	400500	420500
Sedan-2	400510	420500	420500	422500
Sedan-3	280500	290500	300500	280500

Selecting the Number of Dimensions

Every cube has at least two dimensions and a maximum of 256 dimensions. For example, a two-dimensional cube is best suited as a lookup table when you want to calculate values in other cubes that have more dimensions. For example, you can convert the local currency amounts to a reporting currency by using a two-dimensional cube that stores the exchange rates. You retrieve the rate using the TM1 rule.

The number of dimensions in a cube depends mostly on the *dimensionality* of your data. Consider the accounts in a Profit and Loss Statement.

Profit and Loss Statement (in thousands) **Year Ending 31 Dec, 2002**	
Net sales	200,000
Direct costs	35,000

© Copyright IBM Corp. 2007, 2017

Profit and Loss Statement (in thousands) **Year Ending 31 Dec, 2002**	
Direct labor	50,000
Gross Profit	115,000
Salaries	30,000
Payroll	3,500
Electricity	5,000
Rent	10,000
Depreciation	6,000

If you want to examine how the revenue and expenses vary by their factors, you would need to divide the accounts into two groups.

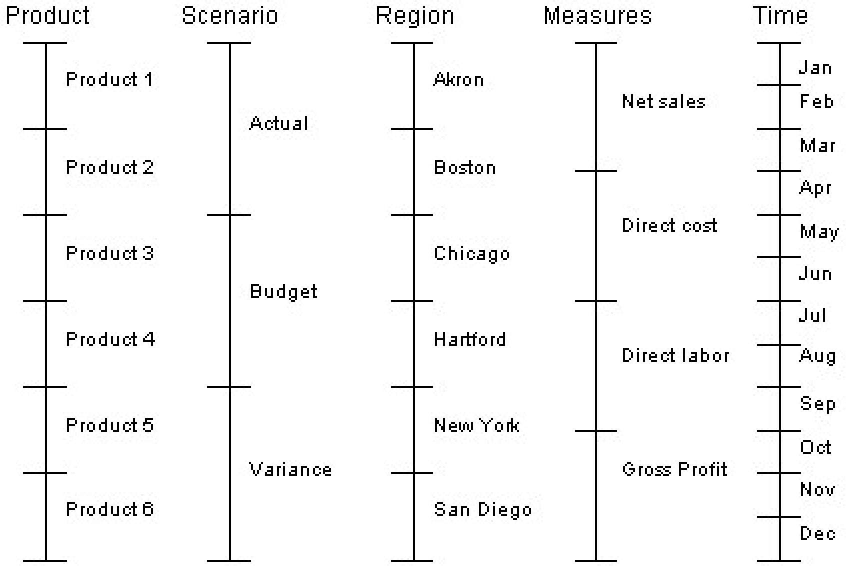
- **Accounts above the Gross Profit line**, such as Net sales and Direct costs, which you can dimension by product, region, scenario (Actual versus Budget), and months.
- **Accounts below the Gross Profit line**, such as Payroll, Electricity, and Rent, which you can dimension by region, scenario (Actual versus Budget), and months, but not product. You cannot directly attribute the overheads to products, so you cannot analyze them at the same level of detail.

This difference in dimensionality suggests two cubes:

- Five-dimensional cube for the accounts above the Gross Profit line
- Four-dimensional cube for the accounts below the Gross Profit line

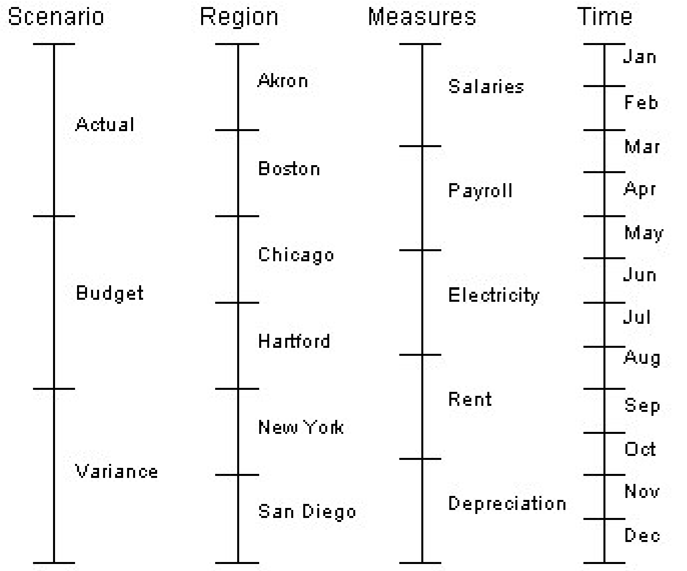
Five-dimensional Cube

The following diagram represents the dimensions and elements in the five-dimensional cube. Each dimension is represented by a vertical line segment. The elements within the dimension are represented by unit intervals.



Four-dimensional Cube

The following diagram represents the dimensions and elements in the four-dimensional cube. Each dimension is represented by a vertical line segment. The elements within the dimension are represented by unit intervals.

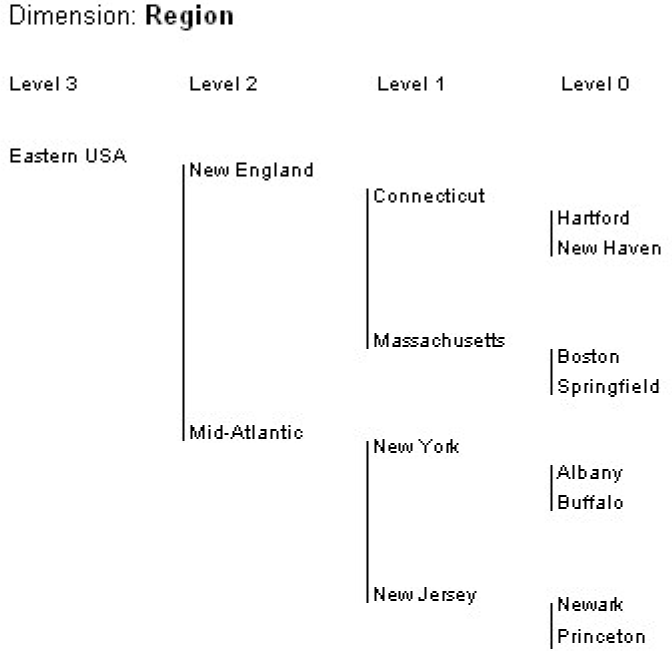


Consolidating Detail Using Dimension Hierarchies

The data you import into a cube provides a snapshot of your business at a specific level of detail. For example, you might import the weekly or monthly sales data for products by city. The dimension elements that identify these data points are simple or leaf-level elements in each dimension: sales for one week, one product, one city.

By using dimension hierarchies, you can easily aggregate *numeric* data into categories that are meaningful in your analyses. Each category corresponds to an aggregation of detail for two or more elements in a dimension. For example, you can create quarterly elements that sum monthly sales amounts. In TM1, elements that represent aggregations are called consolidated elements or consolidations.

The following diagram shows three levels of consolidation for elements of a Region dimension. The cities provide the lowest level of detail (Level 0). The cities roll up into state consolidations (Level 1), which roll up into regional consolidations, which finally roll up into the Eastern USA consolidation (Level 3).



Navigating through a Dimension Hierarchy

A dimension hierarchy provides a navigation path for analyzing data at different levels of detail. Drilling down means navigating to greater levels of detail along one dimension. For example, as you drill down from New England in the above Region dimension, you access the underlying data for two states and then four cities. Drilling up means navigating to summary levels in dimensions.

Using Weights to Express Consolidations

Weight factors determine the contribution of an element to a consolidation. To express that the Connecticut sales total is a sum of Hartford and New Haven, you assign a default weight factor of 1.0 to both Hartford and New Haven.

You can also consolidate elements by subtracting the values associated with the elements. For example, you can express Net Profit as Market Value - Acquisition Value. You would assign a weight factor of 1.0 to Market Value and -1.0 to Acquisition Value. The following table presents four weighting examples of consolidation.

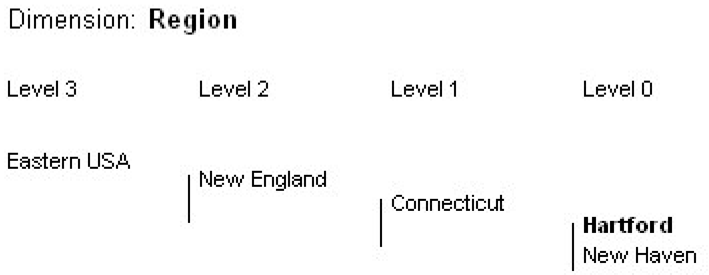
Dimension	Consolidated Element	Consolidation Method	Subordinate Elements	Weight Factors
Account	Net Profit	Subtraction	Market Value Acquisition Value	1.0 -1.0
Month	1Quarter	Addition	January February March	1.0 1.0 1.0
Period	Yearly Budget	4-4-5 Distribution	January February March	.07692 .07692 .09615
Region	Connecticut	Addition	Hartford New Haven	1.0 1.0

Creating Multiple Roll-ups in a Dimension

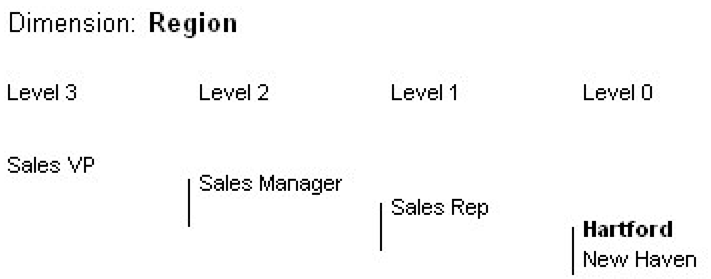
You can roll up low-level numeric data, such as sales and units, in several ways by creating multiple hierarchies in a dimension. By creating multiple roll-ups in a dimension, you can reduce the number of dimensions, and the number of empty cells in a cube.

In the following example, Hartford, a simple element in the Region dimension, rolls up along two paths: geographic and management.

Geographic Roll-up



Management Roll-up



Creating Complex Calculations

TM1 aggregates the consolidations that you create within dimensions by the weight factors you assign. To create ratios between the elements or multiply the values that are associated with the elements, you must create a complex calculation, called a rule.

The following two calculations require TM1 rules:

- Gross Margin = (Gross Profit Net Sales) x 100
- Sales = (Price x Units)

You can also use rules to tally the elements. However, rules take longer to process than consolidations, especially in large or sparse cubes. Sparse cubes have a high percentage of empty cells.

IBM® Cognos® TM1® also supports rule-aware aggregate processing. This feature is visible primarily on IBM® Cognos® Business Intelligence® (BI) clients. In previous versions of Cognos TM1, the default aggregation could not be computed by the Cognos BI server when applied to Cognos TM1 rule-calculated cells. In previous releases, Cognos BI reports returned error cells as a result of default aggregation that is applied to Cognos TM1 rule-calculated cells. With rule-aware aggregation in place, the aggregation numbers are reported based on the semantics of the Cognos TM1 rules.

The following limitations are still in place when you compute default aggregation against rule-calculated cells:

- Multi-dimensional tuple sets cannot be aggregated against calculated cells unless they are a result of cross-join of single dimension member sets. In this case, the aggregation result continues to be an error cell.
- Rule-aware aggregation cannot be applied to MDX calculated members, which means calculated members are not supported in the aggregated member set, nor in aggregation context. In this case, the aggregation over calculated cells still results in an error cell.
- Aggregated cells must be associated with the same rule so that the system can reuse this rule for the aggregation result. The scope of the rule needs to be general enough to include UDC (Use Defined Consolidation) elements that belong to the dimensions of the aggregated member set.

For a comprehensive explanation of rules, see *TM1 Rules* .

Types of Elements

So far, you've learned about simple or base-level elements that you can roll up to consolidations. TM1 supports three types of elements.

Element	Description
Numeric	Identify the lowest-level detail in a dimension. In a cube that contains only numbers, TM1 defines all the lowest-level elements as numeric.
Consolidated	Aggregations of lower-level detail. For example, you could use the 1Quarter element in a time dimension to sum the sales amounts for the first three months of the year.
String	Stores text strings in cells. To include a string in a cell in a cube, the element from the last dimension defining the cell must be a string element. TM1 treats string elements that occur in any dimension other than the last one as numeric elements. The typical use for string elements is in a two-dimensional cube that converts codes in an input file to formal element names. For example, if you want to convert account codes to account names.

Element Attributes

The elements identify data in a cube, and the element attributes describe the elements themselves.

You can use attributes to:

- List features of elements. For example, the square footage of a store and the engine size of a car model.
- Provide alternative names, or aliases. For example, descriptive names of general ledger accounts and the local language versions of product names.
- Control the display format for the numeric data. Typically, you select a display format for the measures you track in a cube.

You can select elements by attribute value in the **Subset Editor**. You can also display element names in TM1 dialog boxes using their aliases.

To create attributes and assign attribute values, use the **Attributes Editor**.

Descriptive Attributes

The following table shows sample attributes that describe car models.

Elements	**Attributes**		
	Horsepower (Numeric)	**Engine (Text)**	**Audio (Text)**
Sedan 1	190	V-8	Compact Disc
Sedan 2	140	Inline 4	Cassette / Radio
Sedan 3	120	Inline 4	Cassette / Radio
Sedan 4	180	V-8	Compact Disc
Sedan 5	140	Inline 4	Cassette / Radio

Alias Attributes

The following table shows German, Spanish, and French versions of English furniture names.

Elements	**Alias Attributes**		
	Deutsche	**Español**	**Français**
Chair	Stuhl	Silla	Chaise
Desk	Schreibtisch	Escritorio	Bureau
Lamp	Lampe	Lámpara	Lampe

Display Format Attributes

The **Cube Viewer** window displays numeric data in the formats that are shown in the following table.

Format Name	**Description**	**Example**
Currency	Numbers appear with a currency symbol and the specified number of decimal places (Precision). TM1 uses the currency symbol that is defined in the Microsoft Windows Regional Settings dialog box.	$90.00
General	Numbers appear with a specified number of decimal places (Precision).	-90

Format Name	Description	Example
Percentage	Numbers appear as percentages, with a specified number of decimal places (Precision).	90.00%
Scientific	Numbers appear in exponential form, with a specified number of decimal places (Precision).	9.0e+001
Date	Numbers appear as a date string. 1=January 1, 1900. There are a number of date formats available. When an element is formatted as Date, and the element is viewed in TM1 Web or TM1 Application Web, you can use a calendar date selector to pick a new date value.	March 31, 2002
Time	Numbers appear as a time string. There are a number of time formats available.	19:53:30 A
Comma	Places commas in the appropriate places in large numbers.	1,000,000
Custom	A user-defined format.	Custom

Using the **Attributes Editor** window, you can select a display format for every element in each dimension in a cube. However, it is recommended that you select display formats only for one dimension, the measures you track in a cube. You can also select a format in the **Cube Viewer** window that applies to cells whose elements do not have a display format defined.

TM1 determines which display format to use in the **Cube Viewer** window, as follows:

1. TM1 first checks the elements in the column dimension for display formats. If dimensions are stacked, TM1 checks from the bottom upward.
2. If no format is found, TM1 checks the elements in the row dimension for display formats. If dimensions are stacked, TM1 checks from right to left.
3. If no format is found, TM1 checks the title elements for display formats. The elements are inspected from right to left.
4. If no format is found, TM1 applies the format for the current view.

   To ensure that TM1 applies the format for the cube measures, position the dimension that containis the measures as the bottommost column dimension.

Setting Display Formats for Rows or Columns
You might want to format the numbers in a single column or row. For example, numbers in a column or row that contain two decimal places might be better represented as whole numbers with no decimal points. To format the numbers in a single column or row, use the **Attribute Editor**.

The Month dimension displays in the column of the view. Any display format you assign to a column dimension overrides the display format you select for the row dimension.

For the first task, let's make sure that no formatting is applied to the Year element of the Month dimension. Follow these steps.

Procedure

1. Open the **Format** view of SalesPriorCube.
2. In the Server Explorer, expand the SalesPriorCube cube so that you can see its dimensions.
3. Right-click the Month dimension, and select **Edit Element Attributes**.

   The **Attribute Editor** opens.

Note that there are no Format attribute values for the Month dimension. You can now be sure that any formats you set for the row dimension will take precedence in the Cube Viewer (or In-Spreadsheet Browser).

4. Click the **Cancel** to close the Attribute Editor.

   For this task, follow the steps to format the numbers in the Units row as whole numbers with zero (0) decimal places.

5. In the Server Explorer, right-click the **Account1** dimension, and select **Edit Element Attributes**.

   The **Attribute Editor** opens.

6. Click the cell at the intersection of the Units element row and the Format column.
7. Click the **Format** button.

   The **Number Format** dialog box opens.

8. Select the **Comma** category, enter **0** in the **Precision** box, and click **OK**.
9. Click **OK** in the **Attributes Editor** dialog box.
10. Click **Recalculate** to recalculate the **Format** view.

    The **Units** values now display as whole numbers without decimal points.

Attributes versus Elements

When you want to list multiple attributes values for a single element, consider creating additional elements or additional dimensions. For example, the exterior color is an attribute of car models. The red models often outsell the other color models. If you create one element per car and another dimension with elements for each color, you can use TM1 to track car sales by color. If you combine sales into a single model, you might lose valuable detail.

Consider another example. In the car models table, there is an attribute category for engine configuration. Each car has a single engine configuration, such as V-8. If any sedans are available in more than one engine configuration, consider creating one element per engine configuration.

Designing Cubes - Summary

Here are guidelines to use when you design cubes:

Procedure

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. Determine how you want to organize or dimension the measures. 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 dimension (actual versus budget)?
3. Determine how you want to consolidate the dimension elements.
4. Create a list of attributes you want to associate with the elements of the cube. Examples of attributes include store square footage, customer IDs, and local language versions of element 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.

Creating Dimensions

When you create a dimension, you identify the leaf-level elements that comprise the dimension and, optionally, any hierarchies (consolidations) within the dimension.

There are four ways to create dimensions:

- **Dimension Editor** - Add elements as well as create and rearrange consolidations while you design dimensions. For more information, see "Creating Dimensions Using the Dimension Editor Window" on page 15.

- **TurboIntegrator** - Import element names from an ASCII, ODBC, cube view, or dimension subset source. Simultaneously create multiple dimensions and establish consolidation within those dimensions. For more information, see *TM1 TurboIntegrator*.

  Note: TM1 requires DataDirect drivers to access an Oracle ODBC source on Solaris or AIX®. These drivers are not supplied with TM1 and must be acquired separately.
- **Importing Data into a New Cube** - Use TurboIntegrator to map input rows from a data source to a cube. Then identify the input columns that supply the cell values and the elements that identify the cell location. For more information about this process, see *TM1 TurboIntegrator*.
- **Dimension Worksheets** - Use these modified Microsoft Excel worksheets to list the elements and hierarchical relationships for one dimension. For more information, see "Creating Dimensions Using Dimension Worksheets" on page 21.

Creating Dimensions Using the Dimension Editor Window

This section walks you through the steps for creating a simple Area dimension using the **Dimension Editor** window. Assume that the hierarchy for the Area dimension includes the consolidated New England element and three simple elements, Connecticut, Massachusetts and Vermont.

Procedure

1. In the **Tree** pane of the Server Explorer, select **Dimensions** beneath the server that you want to contain the dimension.
2. Click **Dimensions**, **Create New Dimension**.

   The **Dimension Editor** opens.

   You can now add elements to the dimension.
3. Click **Edit**, **Insert Element** or click **Insert Sibling** .

   The **Dimension Element Insert** dialog box opens.
4. To add a consolidated element, do the following:

   - Type **New England** in the Insert Element Name field.
   - Select **Consolidated** from the **Element Type** list.
   - Click **Add**.
   - Click **OK**.

   New England now appears as the first element of the dimension, which is a consolidated element. Now let's add three child elements to the New England element. As a result, New England becomes the parent element of the three child elements.
5. Select the **New England** element.
6. Click **Edit**, **Insert Child** or click **Insert Child** .

   The **Dimension Element Insert** dialog box opens. TM1 displays a Parent name of New England, indicating that any elements you create will be the children of New England.
7. In the Insert Element Name field, type **Connecticut** and click **Add**.
8. In the **Insert Element Name** field, type **Massachusetts** and click **Add**.
9. In the **Insert Element Name** field, type **Vermont** and click **Add**.

   The dialog box now contains three children of New England, each with a default weight of 1.
10. Click **OK**.

    The **Dimension Editor** shows the new elements as children of New England.
11. Click **Dimension**, **Save** or click **Save** .

    The **Dimension Save As** dialog box opens.
12. Enter a dimension name and click **Save**.

Dimension names can have a maximum of 256 characters. Always use descriptive dimension names.

The new dimension displays in the list of dimensions on the server.

Modifying a Dimension

After creating a dimension, you can make the following modifications:

- Add siblings to existing elements.
- Add children to existing elements.
- Rearrange the hierarchy structure, such as repositioning elements within consolidations.
- Delete elements from the dimension.
- Delete elements from consolidations.
- Edit element properties, such as changing the weight of an element within a consolidation.
- Rearrange the order of elements in the dimension.

Adding Siblings to Existing Elements

Follow these steps to add siblings to an existing element in the Dimension Editor.

Procedure

1. Right-click the element to which you want to add siblings and select **Edit Dimension Structure**.
2. Click **Edit**, **Insert Sibling**.

   The **Dimension Element Insert** dialog box opens.
3. Enter the name of the first sibling in the **Insert Element Name** field.
4. If applicable, enter an Element Weight.

   The default element weight is 1.
5. Click **Add**.
6. Repeat steps 3 through 5 for each sibling you want to add.
7. Click **OK**.

   TM1 adds the new elements as siblings of the element you selected in step 1.

Adding Children to Existing Elements

Follow these steps to add children to existing elements in the Dimension Editor.

Procedure

1. Right-click the element to which you want to add siblings and select **Edit Dimension Structure**.

   If you add children to a simple (leaf-level) element, the element automatically becomes a consolidated element.
2. Click **Edit**, **Insert Child**.

   The **Dimension Element Insert** dialog box opens.
3. Enter the name of the first child in the **Insert Element Name** field.
4. If applicable, enter an Element Weight.

   The default element weight is 1.
5. Click **Add**.
6. Repeat steps 3 through 5 for each child you want to add.
7. Click **OK**.

   TM1 adds the new elements as children of the element you selected in step 1.

Rearranging the Dimension Hierarchy

Follow these steps to change the position of elements within the dimension hierarchy.

Procedure

1. In the **Dimension Editor**, select the elements you want to move.
   - To select a single element, click the element.
   - To select multiple adjacent element, click the first element, hold down SHIFT, and click the last element. You can also press CTRL-A to select all visible elements.
   - To select multiple non-adjacent elements, hold down CTRL, and click each element.
2. Drag and drop the elements to their new location in the dimension hierarchy.

   As you drag the elements, the cursor changes to indicate where TM1 will drop the elements. Also, the status bar displays a message indicating where TM1 will drop the elements.

   You can also cut and paste elements to rearrange the dimension hierarchy.

Deleting Elements from a Dimension

Follow these steps to delete elements from a dimension.

Procedure

1. Select the elements you want to delete.
   - To select a single element, click the element.
   - To select multiple adjacent element, click the first element, hold down SHIFT, and click the last element.
   - To select multiple non-adjacent elements, hold down CTRL, and click each element.
   - To select elements by hierarchy level, attribute value, or spelling pattern, see *TM1 Perspectives, TM1 Architect, and TM1 Web*.
2. Click **Edit**, **Delete Element**.

   A confirmation dialog box displays that lists the dimension name and asks if you are sure you want to delete the object that you have selected. Click **Yes** to proceed with the deletion, click **No** or **Cancel** to cancel the deletion.

Deleting Elements from a Consolidation

Follow these steps to delete elements from a consolidation.

Procedure

1. Select the elements you want to delete.
   - To select a single element, click the element.
   - To select multiple adjacent element, click the first element, hold down SHIFT, and click the last element.
   - To select multiple non-adjacent elements, hold down CTRL, and click each element.
2. Click **Edit**, **Delete Element from Consolidation** or click **Delete** ☒.

   A confirmation dialog box displays that lists the dimension name and asks if you are sure you want to delete the object that you have selected. Click **Yes** to proceed with the deletion, click **No** or **Cancel** to cancel the deletion.

   TM1 deletes the element from the consolidation, and keeps any other instances of the element in the dimension.

   Note: If you define the element only within the consolidation, TM1 deletes the element from the dimension as well.

Editing Element Properties

You can edit the element properties to assign a new weight to an element of a consolidation, or to change the element type of a leaf-level element.

Note: You cannot change the element type of consolidated elements, and you cannot assign an element weight to any instance of an element that is not a member of a consolidation.

Procedure

1. Select the element.
2. Click **Edit**, **Element Properties**.

The **Dimension Element Properties** dialog box opens.

3. If necessary, select a new element type from the **Element Type** list.
4. If necessary, enter a new Element Weight.
5. Click **OK**.

Setting the Order of Elements in a Dimension

TM1 lets you set the order of elements in a dimension to determine the index value for each element in a dimension. The first element in a dimension has an index value of 1, the second element has an index value of 2, and so on.

Set the order of elements in a dimension is an important feature because many TM1 functions (worksheet, rules, and TurboIntegrator) reference the element index values.

Note: If you change the order of elements in a dimension, any functions that reference element index values return new and possibly unexpected values.

Procedure

1. Order the elements as you want them to appear in the dimension.

   You can use the sort options and drag-and-drop functionality of the **Dimension Editor** to alter the order of elements.
2. Click the **Set Dimension Order** button.
3. Click **Dimension**, **Save**.
   - When the sorting property of the dimension is set to Automatic, TM1 prompts you to change the sorting property to Manual.
   - When the sorting property of the dimension is set to Manual, TM1 inserts any elements you added to the dimension wherever you manually positioned them in the **Dimension Editor**.
4. Click **Yes** to save the new dimension order and set the dimension sorting property to Manual.

   You can set the order of elements even when the Dimension Editor displays only a subset of all dimension elements. For example, if you have a large dimension, you might want to alter and set the order of just a few elements. Be aware that when you set the order of elements with just a subset of elements displayed in the **Dimension Editor**, the entire dimension is affected.

   The following example shows how setting the order of elements when working with a subset affects the entire dimension in Subset Editor.

   For simplicity, this sample dimension contains ten elements with single-letter names, but the concept illustrated in this example applies to larger, more complex dimensions.

   - a
   - b
   - c
   - d
   - e
   - f
   - g
   - h
   - i
5. Now if you select the elements **c**, **d**, and **g** then click **Edit**, **Keep**, the Dimension Editor contains only the selected subset of elements.
6. Now, you decide that you want to change the order of these three elements. You want **d** to be the first element, and **c** to be the last element.
7. Now that the elements appear in the order you want, click **Set Dimension Order** .

   The order of elements for the entire dimension is now set. If you look at the entire dimension, you see that it opens in Dimension Editor as follows.

   - i

- a
- b
- d
- g
- c
- e
- f
- h
- i

For the example, when you set the order of elements for a subset, the new order affects the entire dimension in the following way:

- The subset of elements that was active when the element order was set appear with shaded icons.
- The first subset element maintains its position in the dimension *relative to its nearest predecessor*.
- In the example, element **d** is the first element in the subset when the order of elements was set.
- Element **b** is the nearest predecessor, *exclusive of subset elements*, to **d** in the dimension, so **d** now follows **b** in the dimension structure.
- The other subset elements appear in the dimension structure maintaining their position relative to the first element in the subset.

Setting the Order of Dimension Elements from the Server Explorer
You can also set the order of dimension elements from the Server Explorer without opening the **Dimension Editor**. You can also select a sorting property for the dimension from these three automatic sort orders:

- Name
- Level
- Hierarchy

After you set the sorting property, TM1 inserts the elements you added to the dimension according to their position within the sort order. For example, if you set an automatic sort order of Name, TM1 inserts the new elements into the dimension in alphabetical order.

Procedure

1. Right-click the dimension in the Server Explorer.
2. Click **Set Elements Order** .

   The **Dimension Element Ordering** dialog box opens.
3. Select a sort type.

Sort Type	Description
Automatic	Enables the Automatic Sort By options: Name, Level, and Hierarchy.
Manual	Orders elements as they currently exist in the dimension structure and sets the dimension sorting property to Manual.

4. If you select the **Manual** sort type, skip to step 7.
5. Select an **Automatic Sort By** option.

Sort By	Description
Name	Sorts elements alphabetically
Level	Sorts elements by hierarchy level.
Hierarchy	Sorts elements according to the dimension hierarchy.

6. If applicable, select a **Sort Direction**.

7. Click **OK**.

Results
You have now set the order of the dimension elements. When you open the dimension, you will see the elements in order according to the **Sort By** option you specified in step 5.

Managing the Display of Elements in the Dimension Editor

The Dimension Editor includes several features that let you manage the way elements display. For example, when you work with large dimensions it might be helpful to display only the elements of a certain hierarchy level, or you might want to view the elements in alphabetical order.

When you alter the display of elements in the **Dimension Editor,** you do not change the dimension structure, you simply change the way TM1 presents the elements in the window.

The upcoming sections describe how to manage the display of elements in the Dimension Editor.

Keeping Elements

Follow these steps to display only the selected elements in the Dimension Editor.

Procedure

1. Select the elements you want to display.
   - To select a single element, click the element.
   - To select multiple adjacent element, click the first element, hold down SHIFT, and click the last element.
   - To select multiple non-adjacent elements, hold down CTRL, and click each element.
2. Click **Edit**, **Keep** or click **Keep**.

   The **Dimension Editor** displays only the elements you selected.

Hiding Elements

Follow these steps to hide selected elements in the Dimension Editor.

Procedure

1. Select the elements you want to hide.
   - To select a single element, click the element.
   - To select multiple adjacent element, click the first element, hold down SHIFT, and click the last element.
   - To select multiple non-adjacent elements, hold down CTRL, and click each element.
2. Click **Edit**, **Hide** or click **Hide**.

   The Dimension Editor hides the elements you selected. All other elements remain visible.

Sorting Elements Alphabetically

You can sort the elements in the Dimension Editor in ascending or descending alphabetical order.

Sort Order	Description
Ascending Alphabetical	Click **Edit**, **Sort By**, **Ascending** or click **Sort Ascending**
Descending Alphabetical	Click **Edit**, **Sort By**, **Descending** or click **Sort Descending**

Sorting Elements by Index Value

You can sort the elements in the **Dimension Editor** in ascending or descending order according to the index value.

Sort Order	Description
Ascending Index Value	Click **Edit**, **Sort By**, **Index Ascending** or click **Sort By Index, Ascending**
Descending Index Value	Click **Edit**, **Sort By**, **Index Descending** or click **Sort By Index, Descending**

Sorting Elements by Hierarchy

You can also sort elements as they appear in the dimension hierarchy.

Sort Type	Description
As they appear in the dimension hierarchy	Click **Edit**, **Sort By**, **Hierarchy** or click **Hierarchy Sort**

Viewing Elements by Alias

If you defined an alias attribute for a dimension, you can view the elements by their aliases in the **Dimension Editor**.

The dimensions in the TM1 Sample Data directory have aliases defined for French and German, so you can view all element names by their foreign language equivalents.

Procedure

1. Open the Month dimension in the Dimension Editor.
2. Click **View**, **Toolbars**, **Alias** to display the **Alias** toolbar.

   The **Alias** toolbar contains two objects: a **Use Aliases** toggle button to turn the display of aliases on or off, and a **Select Alias** list from which you can select an alias.
3. Select **Monat** (German for 'Month') from the **Select Alias** list.
4. Click the **Use Aliases** button.

Results

By default, TM1 does not display the aliases. The **Dimension Editor** now displays all elements by their German aliases.

Creating Dimensions Using Dimension Worksheets

A dimension worksheet is a modified Microsoft Excel spreadsheet in which you list elements and hierarchical relationships for one dimension. TM1 saves the worksheet data in two files: dimension worksheet (*dimname*.xdi) and compiled file (*dimname*.dim).

- When you create a dimension using the Dimension Editor window or TurboIntegrator, TM1 writes only to the *dimname*.dim (compiled) file.
- When you modify a dimension using the Dimension Editor, TM1 saves your changes to the .dim file. If an .xdi file exists for the dimension, TM1 prompts you to save changes to the dimension worksheet. If you decline to update the dimension worksheet, the dimension structure in the .dim file will differ from that in the .xdi file.
- To ensure that TM1 has access to all of your dimension changes, create and maintain dimensions with dimension worksheets *or* the Dimension Editor. Do not mix methods. This can lead to data loss.

Keeping Your Data Synchronized

TM1 saves the dimension worksheet files to the first directory listed in the Local Server Data Directory field in the Options dialog box. Therefore, TM1 might save your dimension worksheet (.xdi) to a different directory than your dimension file (.dim).

Note: Be very careful when editing dimensions using worksheets. There are several ways in which data can be lost if you do not take proper precautions. The following examples show two ways you can lose dimension data.

- On Monday, you edit a dimension worksheet for the Account1 dimension, and save the dimension. On Tuesday, another administrator uses the Dimension Editor to make changes to the dimension. The changes she makes are not propagated to the dimension worksheet file. On Wednesday, you make another change to the dimension using the out-of-date dimension worksheet. Your changes overwrite the changes made by the other administrator.

- Two administrators decide to update a dimension on the same server using two different dimension worksheets. This is very dangerous. TM1 can overwrite the changes very easily. *Use one set of dimension worksheets. Try to avoid having more than one .xdi file for any dimension.*

To avoid confusion and possible data loss, use precautions when editing dimension worksheets. We recommend using one of the following procedures.

- Use the remote server data directory to store the dimension worksheets.
- Use a special directory to store your dimension worksheets.

Using the Remote Server Data Directory
To edit the dimension worksheets in the server data directory on the remote server, follow these steps.

Procedure

1. Set the Local Server Data Directory in the **Options** dialog box to the data directory used by your remote IBM Cognos TM1 server.
2. Clear the option **Connect to Local Server on Startup**.
3. Verify that any local servers running on your computer are shut down.
4. Log in to the remote TM1 server.
5. Make all of the modifications to your dimensions using the worksheet files in the remote server data directory.
6. Click **TM1** > **Dimension Worksheets** > **Save** in Excel.

   TM1 writes both the .dim file and the .xdi file to the remote server data directory.

Using a Worksheet Directory
By using a special worksheet directory, you can sharply restrict access to your dimension worksheets. This can be very useful in security-conscious installations. To set up a worksheet directory, follow these steps.

Procedure

1. Create a worksheet directory somewhere on your file system.
2. Move all of your worksheet files (.xdi for dimensions, and .xru for rules) to the worksheet directory.
3. Set the Local Server Data Directory in the **Options** dialog box to the worksheet directory.
4. Connect to the remote server that contains the dimension you want to update, and any other servers you want.
5. Make all of the modifications to your dimensions using the worksheet files in the worksheet directory.
6. Click **TM1** > **Dimension Worksheets** > **Save** in Excel.

   The **Select Server Name** dialog box opens.
7. Select the server on which you want to save the compiled dimension.
8. Click **OK**.

Creating Dimension Worksheets
Follow these stesps to create a dimension using dimension worksheets.

Procedure

1. Click **TM1** > **Dimension WorkSheets** > **New** in Excel.

   The **Create a Dimension** dialog box opens.
2. In the top box, enter a name for the dimension as follows:

   - To create a dimension on your local server, type the name of the dimension. For example: Product.
   - To create a dimension on a remote server, type the server name, a colon, and then the dimension name. For example, sales:Product creates the Product dimension on the sales server.

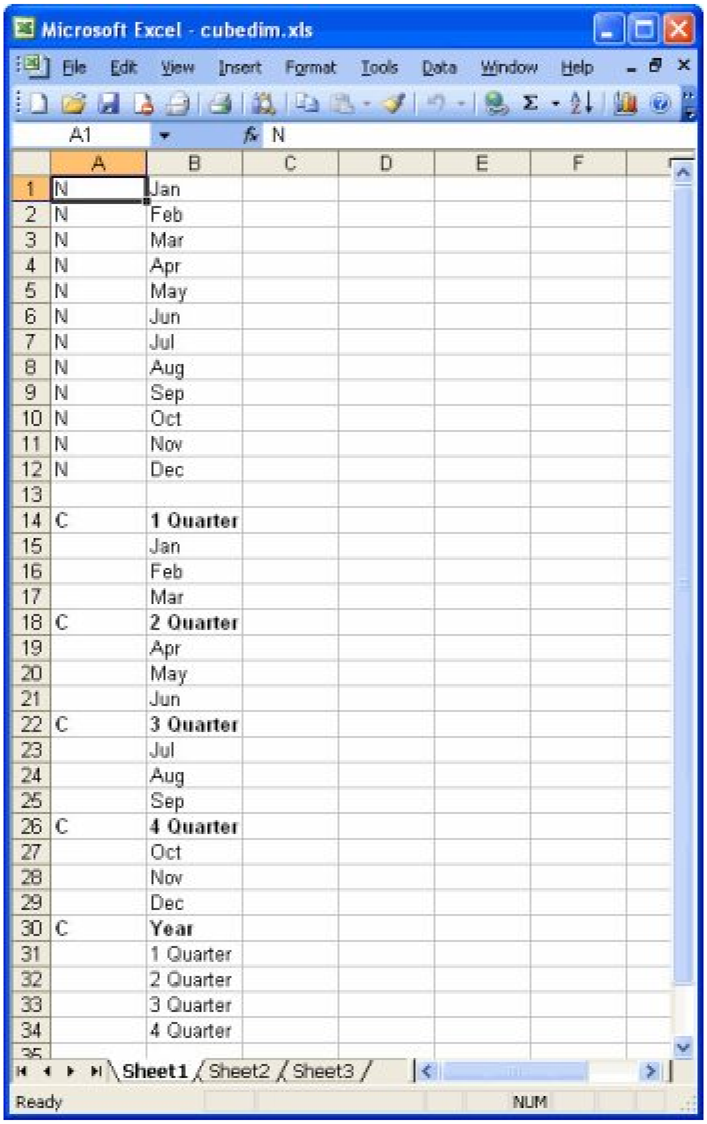
   Note: You must be the TM1 administrator to create a dimension on a remote server.

3. Click **OK**.

   A blank dimension worksheet opens in Excel.

Filling Out Dimension Worksheets

The following worksheet defines the structure for a Month dimension.


Microsoft Excel - cubedim.xls
File Edit View Insert Format Tools Data Window Help
A1
N
Sheet1
Sheet2
Sheet3
Ready
NUM

	A	B	C	D	E	F
1	N	Jan				
2	N	Feb				
3	N	Mar				
4	N	Apr				
5	N	May				
6	N	Jun				
7	N	Jul				
8	N	Aug				
9	N	Sep				
10	N	Oct				
11	N	Nov				
12	N	Dec				
13						
14	C	**1 Quarter**				
15		Jan				
16		Feb				
17		Mar				
18	C	**2 Quarter**				
19		Apr				
20		May				
21		Jun				
22	C	**3 Quarter**				
23		Jul				
24		Aug				
25		Sep				
26	C	**4 Quarter**				
27		Oct				
28		Nov				
29		Dec				
30	C	**Year**				
31		1 Quarter				
32		2 Quarter				
33		3 Quarter				
34		4 Quarter				

The following sections describe how to define simple and consolidated elements in a dimension worksheet.

Defining Simple Elements

Begin filling out the dimension worksheet by defining the dimension's simple (leaf-level) elements, starting in row 1.

Procedure

1. In column A, specify the element type.
   - For numeric elements, type **N**.
   - For string elements, type **S**.

   In the example, all elements are numeric.
2. In column B, type the name of an element.

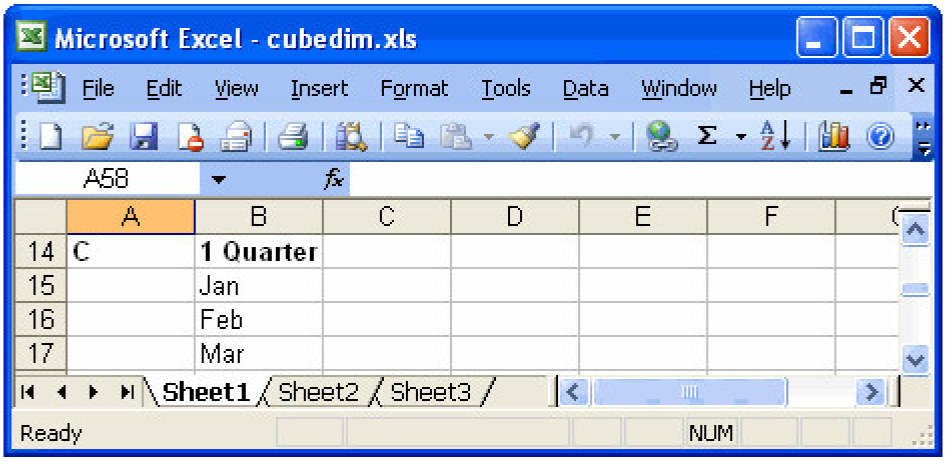
Defining Consolidated Elements

After you define the simple elements in the dimension, you can define the consolidated elements.

Procedure

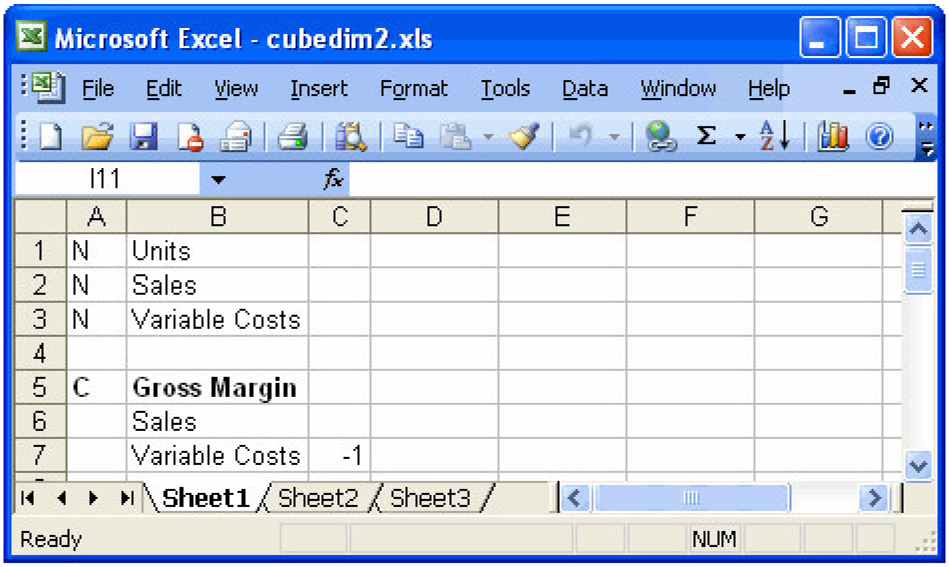
1. In an empty row below the last simple element, type **C** in column A.
2. Type the name of the consolidated element in column B.
3. In subsequent rows, type the names of the consolidated element's children in column B.

   For example, the following excerpt from a dimension worksheet shows the consolidated element 1 Quarter that is defined as the consolidation of the children Jan, Feb, and Mar.


Microsoft Excel - cubedim.xls
File Edit View Insert Format Tools Data Window Help
A58
Sheet1 Sheet2 Sheet3
Ready
NUM

	A	B	C	D	E	F
14	C	**1 Quarter**				
15		Jan				
16		Feb				
17		Mar				

Weighting Elements in a Consolidation
Use column C to weight the elements in a consolidation. To calculate the Gross Margin, for example, you subtract Variable Costs from Sales. To express this calculation, assign the weight -1 to the Variable Costs element in the consolidation. The following worksheet shows the weighting in the Account1 dimension.


Microsoft Excel - cubedim2.xls
File Edit View Insert Format Tools Data Window Help
I11
Sheet1 Sheet2 Sheet3
Ready
NUM

	A	B	C	D	E	F	G
1	N	Units					
2	N	Sales					
3	N	Variable Costs					
4							
5	C	**Gross Margin**					
6		Sales					
7		Variable Costs	-1				

Saving Dimension Worksheets
To save the dimension worksheet and compile the dimension, click **TM1** > **Dimension Worksheets**, **Save** in Excel. TM1 updates the dimension worksheet file (*dimname*.xdi) and creates the compiled dimension file (*dimname*.dim).

Note: Be sure not to use **File**, **Save** in Excel. This action saves only the .xdi file. TM1 does not compile the dimension and save the .dim file.

Using Named Hierarchy Levels with TM1 Dimensions

You can assign your own custom names to the hierarchy levels of the TM1 dimension by using the `}HierarchyProperties` control cube. The named levels you create can then be used when you externally access TM1 data with IBM Cognos Report Studio, MDX statements or other MDX OLAP tools. You can also assign a default member for the dimension.

For example, instead of using the generic hierarchy level names of level000, level001, level002, you could assign names that describe the levels in a Customers dimension as shown in the following table.

TM1 Dimension Levels	Example Named Dimension Levels
level000	All
level001	State
level002	City
level003	Individual

Configuring Named Levels

Use the }HierarchyProperties control cube to configure named levels.

For more details about the }HierarchyProperties control cube, see the section about control cubes in *TM1 Operations*.

Procedure

1. In TM1 Architect, click the**View** menu and select **Display Control Objects**.
2. In the navigation pane, click to expand the **Cubes** node.
3. Double-click the **}HierarchyProperties** control cube.

   The }HierarchyProperties control cube opens.
4. Click the title dimension list to select the dimension for which you want to assign named levels.
5. In the **defaultMember** cell, enter an existing element name to set as the default member for this dimension.

   The element name you enter here may filter the dimension when TM1 data is retrieved from an external application like IBM Cognos Report Studio.

   Enter the name of the top element in the dimension hierarchy so all the dimension elements are retrieved by default.

   For example, set World as the default member for the Region dimension.
6. In the **level000** to **level020** cells, enter your own custom name for each hierarchy level that exists in the dimension.

   Note: Named levels are limited to a length of 255 single-byte characters. For details, see "String Length Limit for Named Levels" on page 25.
7. After configuring named levels, you must do one of the following to apply the changes:
   - Restart the IBM Cognos TM1 server, or
   - Run the `RefreshMdxHierarchy` function in a TurboIntegrator process. For details, see "Using the RefreshMdxHierarchy TurboIntegrator Function with Named Levels" on page 26.

Results

Note: Changes to element names or dimension structure are not automatically detected by the named levels feature. If your dimension changes, you must first manually update the named level assignments in the }HierarchyProperties control cube and then either restart the TM1 server or run the `RefreshMdxHierarchy`TurboIntegrator function to update the MDX hierarchies in the TM1 server.

String Length Limit for Named Levels

Named levels are limited to a length of 255 single-byte characters. Even though the `}HierarchyProperties` control cube supports long strings, MDX statements can return an error with named levels longer than 255 characters.

If you enter a named level with a length greater than 255 single-byte characters, TM1 displays an error when the server starts up:

4648 ERROR 2008-06-27 13:50:04,532 TM1.Hierarchy User-defined level name modified from ...

Using the RefreshMdxHierarchy TurboIntegrator Function with Named Levels

After configuring or editing the named levels in the }HierarchyProperties control cube, use the `RefreshMdxHierarchy` function to update the MDX hierarchies in the TM1 server without requiring you to restart the server.

Procedure

1. Create a new TI process.
2. Enter the `RefreshMdxHierarchy` function on the **ProLog** tab using the following format:

   ```
   RefreshMdxHierarchy(dimensionName)
   ```

   where the optional parameter, *dimensionName*, can either specify a specific dimension to update or can be blank to update all dimensions.

   For example, to update all dimensions:

   ```
   RefreshMdxHierarchy('');
   ```

   To update only the customers dimension:

   ```
   RefreshMdxHierarchy('customers');
   ```

3. Run the TI process.

Using Multiple Hierarchies

IBM Cognos TM1 dimensions can include one or more hierarchies. This section describes the characteristics of multi-hierarchy dimensions, comparisons to single-hierarchy dimensions, and the available functions to manage and leverage multiple hierarchies. Currently, multiple hierarchies can be implemented using TurboIntegrator or Planning Analytics Workspace, but not Dimension Editor or Dimension Worksheets.

Note: By default, multiple hierarchies are not enabled. An administrator must configure the EnableNewHierarchyCreation tm1s.cfg setting before you can work with multiple hierarchies and use the related TurboIntegrator (TI) and Rules functions.

The dimensional modeling approach, wherein all dimensions include a single hierarchy, typically results in a larger number of dimensions per cube. In some cases, "repeating groups" can occur. For example, a ProductCategory dimension includes Product1 in its Commercial category and Product2 in its Retail category. When a value is addressed by Product1, the ProductCategory coordinate is always Commercial. If the product category is a true attribute, then these "repeating groups" are not necessary since a product will never exist in both Commercial and Retail categories.

Alternatively, a multi-hierarchical approach can be used where a Product dimension includes a ByCategory hierarchy, including other hierarchies such as ByPriceRange or ByRegion. This method reduces the number of dimensions in the cube. Added complexity comes with the need to specify the hierarchy that an element pertains to; whether in querying, or in TI or cube rules.

Multiple hierarchies also provide greater flexibility when querying. With a single-hierarchy model, querying against a 10 dimension cube requires all queries to have exactly 10 hierarchies. With multiple hierarchies, queries can have 10 or more hierarchies, as needed. When a query includes more than one hierarchy, an intersection effect occurs. For example,

```
SELECT { [Products].[ByPriceRange].[PriceRange1] } ON ROWS, { [Measures].[Target] } ON
      COLUMNS FROM [Cube]
```

displays the total Target from the PriceRange1 products. We can refine the total by including an extra hierarchy in the query. For example,

```
SELECT { [Products].[ByPriceRange].[PriceRange1] } ON ROWS, { [Measures].[Target] } ON
      COLUMNS FROM [Cube] WHERE ( [Products].[ByCategory].[Retail] )
```

reduces the total by only including PriceRange1 products that appear in the Retail category.

TurboIntegrator and cube rule functions have been added to permit explicit hierarchy specification. Separate functions are available for single and multiple hierarchies. If your cubes contain single-hierarchy dimensions only, you can use either variant. For example, the following two statements are identical.

```
DimensionElementInsert('dimension', '', 'element', 'c');
HierarchyDimensionElementInsert('dimension', 'dimension', 'element', 'c');
```

For single-hierarchy dimensions, the hierarchy shares the same name as the dimension. The second parameter of the HierarchyDimensionElementInsert function is the hierarchy name. Either of the above statements can be used for a single-hierarchy dimension. However, if you have a dimension with a second hierarchy (such as hierarchy2), you must use the multi-hierarchy function and specify the particular hierarchy name. For example,

```

HierarchyDimensionElementInsert('dimension', 'hierarchy2, 'element', 'c');
```

For detailed information on supported TI and cube rule functions, see *TM1 Reference*.

Creating Cubes

You can always create cubes on your local server. You must be the TM1 administrator to create cubes on remote servers.

There are two ways to create cubes:

- **Empty Cube** - Select from a list of existing dimensions in the **Creating Cube** window to create a new cube with no data.
- **External Data Sources** - Use TurboIntegrator to identify and map dimensions and data from an external data sources to a new or existing cube.

This section documents creating cubes in the **Creating Cube** window. For information about creating cubes in TurboIntegrator, see *TM1 TurboIntegrator*.

Ordering Dimensions in a Cube

Dimensions in a cube have an order that you select when you create a cube. The order you select can affect system performance, so you should give some consideration to the order of dimensions before creating a cube.

As a first step toward ordering dimensions, divide the dimensions into two groups: sparse and dense dimensions. A dense dimension has a high percentage of values for its elements. You can estimate the density by answering this question: If one element in the dimension has a value, keeping the elements of the other dimensions constant, what is the probability that the other elements in the dimension have values?

For example, if you have a budget in January for a given account and region, you probably also have a value for the remaining months. Therefore, the Month dimension is probably dense. Similarly, if you have a budget value for a given month, account, and region, you probably also have an actual value, making ActVsBud a dense dimension.

However, in a worldwide sales cube, you probably do not sell every product in every region. Therefore, you would treat Product and Region as sparse dimensions.

We generally recommend that you order the dimensions as follows: smallest sparse to largest sparse, followed by smallest dense to largest dense. However, some flexibility is required. For example, it is probably better to put a very small, dense dimension such as ActVsBud that has only two or three elements before a very large but sparse dimension, such as Product, which might have thousands of elements.

Creating a Cube

Follow these steps to create a cube.

Procedure

1. Open the Server Explorer.
2. In the **Tree** pane, select **Cubes** beneath the server on which you want to create the cube.

3. Click **Cubes**, **Create New Cube**.

   The **Creating Cube** dialog box opens. The **Available Dimensions** box on the left lists the dimensions stored on the server.
4. Type a cube name in the **Cube Name** field.

   Note: If you do not type a name, TM1 names the new cube **Unnamed**.
5. In the **Available Dimensions** box, double-click the name of the dimension you want to use as the *first* dimension in the new cube.

   The dimension name moves to the **Dimensions in new cube** box.

   You can also use the button ➡ to move selected names from the **Available Dimensions** box to the **Dimensions in new cube** box. To select multiple adjacent names, click and drag across the names. To select multiple non-adjacent names, hold down CRTL, and click each name.
6. Repeat the selection process for all the dimensions you want to include in the new cube. You must select at least two dimensions. The maximum number of dimensions is 256.
7. Using the up ⬆ and down ⬇ arrows, rearrange the dimensions if necessary. To remove a dimension from the list, double-click the dimension name.
8. If you want to specify the cube properties, click **Properties**. If you do not want to assign cube properties, skip to step 13.

   The **Cube Properties** dialog box opens.

   From here, you can set a Measures and Time dimension for the cube, and specify if the cube is loaded automatically or on demand.

   Note: OLE DB for OLAP clients may include provisions for referencing Measures and Time dimensions. TM1 does not reference Measures and Time dimensions, but does allow you to set these properties for other OLAP clients that may access the cube.
9. To set a Measures dimension, select a dimension from the **Measures Dimension** list.

10.To set a Time dimension, select a dimension from the **Time Dimension** list.

11.Specify how to load the cube:

- To load the cube into server memory only when a client requests cube data, select the **Load On Demand** box.
- To automatically load the cube into memory when the server starts, clear the **Load On Demand** box.

12.Click **OK** to save the properties and return to the **Creating Cube** dialog box.

13.Click **Create Cube** to create the cube.

The Server Explorer window opens. The new cube displays in alphabetical order in the **Cubes** list in the **Tree** pane.

Optimizing the Order of Dimensions in a Cube

If you're not extremely familiar with your business data, it's possible to specify an order of dimensions during cube creation that results in less than optimal performance. Similarly, it's possible for the distribution of data in a cube to change over time, making the order of dimensions specified during cube creation less than ideal. To address these issues, TM1 includes a feature that lets you optimize the order of dimensions in a cube, thereby consuming less memory and improving performance.

When you optimize the order of dimensions in a cube, TM1 does *not* change the actual order of dimensions in the cube structure. TM1 *does* change the way dimensions are ordered internally on the server, but because the cube structure is not changed, any rules, functions, or applications referencing the cube remain valid.

As you change the order of dimensions, you can instantly view a report detailing the impact your changes have on cube memory consumption.

For the following reasons, you should optimize the order of dimensions in a cube only in a development environment while you are trying to determine optimal cube configuration:

- Significant memory resources are required for the IBM Cognos TM1 server to reconfigure the order of dimensions in a cube. During the re-ordering process, the temporary RAM on the TM1 server increases by a factor of two for the cube that you are re-ordering. For example, a 50 MB cube requires 100 MB of RAM to reconfigure.

- Re-ordering puts a read lock on the server, locking all user requests while the re-order is performed.

Note: You must be a member of the ADMIN group to optimize the order of dimensions in cubes. The optimization option is only available for cubes on remote servers; you cannot optimize the order of dimensions in cubes on a local server. Also, when you optimize the order of dimensions in a cube, you should not move the string dimensions *from* the last position, nor move the string dimensions *to* the last position.

Procedure

1. In the Tree pane of the Server Explorer, select the cube you want to optimize.
2. Click **Cube**, **Re-order Dimensions**.

   The **Cube Optimizer** dialog box opens.
3. Select a dimension in the **New Order of Dimensions** list box.
4. Click the up ⇧ or down ⇩ arrows to change the order of the dimension in the cube.
5. Click **Test**.

   Note the value next to the Percent Change label. If this value is negative, the new order of dimensions consumes less memory and is therefore more efficient.
6. Repeat steps 3 through 5 until you achieve the most efficient ordering of dimensions.
7. Click **OK**.

Editing Cube Properties

TM1 allows you to set cube properties that specify measures and time dimensions used by OLE DB for OLAP applications, and that determine whether a cube loads automatically or on demand. Usually, you set these cube properties when you create a cube, but you can edit the properties any time.

Editing Measures and Time Dimension

OLE DB for OLAP client applications include provisions for measures and time dimensions. Even though TM1 clients do not include these provisions, you can use TM1 to set measures and time dimensions for cubes that you access by OLE DB for OLAP clients.

Procedure

1. Select the cube in the **Tree** pane of the Server Explorer.
2. Click **Cube**, **Properties**.

   The **Cube Properties** dialog box opens.
3. Select a measures dimension in the **Measures Dimension** list.
4. Select a time dimension in the **Time Dimension** list.
5. Click **OK**.

Editing the Load on Demand Property

By default, TM1 loads all cubes into memory when a server starts. While this allows fast access to TM1 data, it can consume significant server resources. If your server contains infrequently accessed cubes, you can conserve resources by setting those cubes to load only when a client attempts to access the cube data.

Procedure

1. Select the cube in the **Tree** pane of the Server Explorer.
2. Click **Cube**, **Properties**.

   The **Cube Properties** dialog box opens.
3. Specify how to load the cube the cube:

   - Select the **Load On Demand** box to load the cube on demand.
   - Clear the **Load On Demand** box to automatically load the cube when the server starts.
4. Click **OK**.

Creating Pick Lists

A pick list is a list of valid values for a specific element or cube cell. When an administrator defines a pick list for an element or a cell, a drop-down menu containing the defined values is available in the specified cell when browsing a cube in any of the TM1 clients.

Values in cells containing a pick list are validated; a user must select one of the predefined values for the cell. If a user attempts to enter a value that is not valid for the cell, an error appears indicating that only values from the pick list can be entered in the cell.

Pick List Usage Notes

You should be aware of the following requirements and behaviors when using pick lists.

- 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.
- When using pick lists with Excel 2007, you *must* have Excel 2007 Service Pack 2 installed if you want to use pick lists in conjunction with automatic calculation mode. If you are running Excel 2007 without Service Pack 2, you should set Excel's calculation mode to manual. Using automatic calculation in the absence of Service Pack 2 can result in access errors on cells containing pick lists.
- Do not use double quotation marks in a pick list value that may be viewed in the TM1 Web or TM1 Application Web Cube Viewer. In TM1 Web or TM1 Application Web, pick list values that contain double quotation marks prevent display of the contents of the pick list and may make the view unusable. Double quotation marks in pick lists work correctly in TM1 Websheets.
- All pick lists in TM1 Web and TM1 Application Web automatically contain a selectable null value. To make null values available in pick lists in TM1 Architect and Perspectives, you must explicitly define a null value in a static pick list. Null values cannot be used in dimension or subset pick lists in TM1 Architect and Perspectives.
- Pick lists in TM1 Web and TM1 Application Web do not support HTML codes for special characters. If you want a special character to appear in a pick list in TM1 Web or TM1 Application Web, you must enter the actual special character when you create the pick list. For example, to include the "greater than" character in a pick list, you must enter **>** when creating the pick list, rather than the HTML code `&gt;`.

Pick List Types

You can create three types of pick lists: static, subset, and dimension.

Static Pick Lists

A static pick list is composed of a comma-delimited list of values using the syntax `static:value1:value2:value3:value4`.

For example, `static:red:orange:yellow:green` results in a pick list containing the values red, orange, yellow, and green.

To include a null value at the beginning or in the middle of a static pick list, use two consecutive colons without intervening characters in the pick list definition. For example, `static::value1:value2::value3:value4` results in a pick list with a null value before `value1`. Similarly, `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 following `value4`.

Subset Pick Lists

A subset pick list contains values corresponding to all elements of a named subset. If the members of the subset change, the values available in the pick list change correspondingly.

A subset pick list is defined using the syntax `subset:dimension_name:subset_name`.

For example, `subset:Products:Winter` results in a pick list containing all elements from the Winter subset of the Products dimension.

Dimension Pick Lists
A dimension pick list contains values corresponding to all elements of a dimension. If the members of the dimension change, the values available in the pick list change correspondingly.

A dimension pick list is defined using the syntax `dimension:dimension_name`.

For example, `dimension:Months` results in a pick list containing all elements from the Months dimension.

Creating Pick Lists with Element Attributes

The easiest way to create a pick list is to define a text attribute named Picklist for a dimension. You can then specify the members of a pick list for each element within the dimension, using any of the pick list types described above. When an element has a pick list defined, any cube cell identified by that element displays a drop-down list containing the pick list values.

Procedure

1. In the Server Explorer, right-click the dimension for which you want to define pick lists, then click **Edit Element Attributes**.
2. In the Attributes Editor, click **Edit**, **Add New Attribute**.
3. In the New Attribute dialog box, enter Picklist as the attribute name.
4. Select **Text** as the attribute type.
5. Click **OK**.

   The Attributes Editor now contains a new column titled Picklist.
6. For each element for which you want to create a pick list, enter a valid pick list definition at the intersection of the element name and the Picklist column.
   a) To enter a static pick list, enter a comma-delimited list of values using the syntax `static:value1:value2:value3:value4`.
   b) To enter a subset pick list, enter the pick list definition using the syntax `subset:dimension_name:subset_name`.
   c) To enter a a dimension pick list, enter the pick list definition using the syntax `dimension:dimension_name`.
7. Click **OK** to close the Attributes Editor and save the pick list definitions.

Creating Pick Lists with Control Cubes

You can also create pick lists with control cubes. This gives you greater control over which cube cells should contain pick lists and allows greater flexibility in defining pick lists for individual cells. 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.

A pick list control cube is composed of the same dimensions as the regular cube it is associated with, along with an additional dimension named }Picklist. The }Picklist dimension contains a single string element, named Value.

Creating a Pick List Control Cube

Use this procedure to create a pick list control cube.

Procedure

In the Server Explorer, right-click the regular cube for which you want to create a pick list control cube, then **Create Pick List Cube.**

A new control cube is created, using the naming convention }Picklist_*cubename*. For example, when you create a pick list control cube for the Orders cube, the control cube is named }Picklist_Orders.

Note: If you cannot view control cubes in the Server Explorer, click **View**, **Display Control Objects** to enable the display of control cubes and other control objects.

Defining Pick Lists for Individual Cells in a Control Cube

Follow these steps to define pick lists for individual cells in a control cube. The pick lists defined in the control cube are used to display pick list values in the associated regular cube.

Procedure

1. Double-click the control cube in the Server Explorer.

   The control cube opens in the Cube Viewer.
2. Configure the view of the control cube as necessary to view the cells for which you want do define pick lists. For details on configuring cube views, see *TM1 Perspectives, TM1 Architect, and TM1 Web*.
3. In each cell for which you want to create a pick list, enter a pick list definition. You can enter any of the pick list types in the control cube: static, subset, or dimension.
4. Click **File**, **Recalculate** to recalculate the cube view.

Using Rules to Define Pick Lists In a Control Cube

Rules that define pick lists follow the same conventions as all other TM1 rules. A rule statement that defines a pick list must include an area definition (the portion of the cube to which the rule applies), a string qualifier, and a formula. For pick list rules, the formula is the pick list definition that you want to apply.

Also, when multiple rule statements apply to overlapping areas, the statements should be ordered from most restrictive area to least restrictive area.

For more information on creating rules, including details on specifying an area definition, see *TM1 Rules*.

Procedure

1. In the Server Explorer, right-click the }Picklist control cube for which you want to create a rule, then click **Create Rule**.

   The Rules Editor opens.
2. Using a standard rules area definition, specify the cells you want the pick list to appear in.
3. Immediately after the area definition, type =`S:`. This is the string qualifier, indicating that the rule applies to string cells.
4. Immediately after the string qualifier, enter a pick list definition, enclosed in single quotes, then enclosed in parentheses. For example, `('static:spring:summer:winter:fall')`.
5. Immediately after the pick list definition, type a semi-colon (;) to terminate the rule statement.

   Using the procedure described in these steps should result in a rule statement similar to the following example, which indicates that an cell identified by the fabric element will display a static pick list containing the values wool, cotton, silk, and nylon.

   ```
   ['fabric']=S:('static:wool:cotton:silk:nylon');
   ```

Other Pick List Rules Examples

The following examples illustrate rules statements that define pick lists.

Rule statement	Description
`['size','shirts']=S:('static:16:17:18');`	This rule statement indicates that any cell identified by the elements **size** and **shirts** will display a static pick list composed of the values 16, 17, an 18.

`['size',{'sweaters','vests','jackets'}]=`
`S:('static:XS:S:M:L:XL');`
	This rule statement indicates that any cell identified by the element **size** and *any* of the elements **sweaters**, **vests**, or **jackets** will display a static pick list composed of the values XS, S, M, L, and XL.

`['fabric']=S:('dimension:materials');`	This rule statement indicates that any cell identified by the element **fabric** will display a pick list composed of all elements in the materials dimension.

Excluding Cells from Pick Lists

There may be some circumstances when you do not want an individual cell or a specific area of a cube to use pick lists. To prevent a cell from displaying a pick list, enter none in the appropriate pick list control cube cell or use `('none')` as the formula in a rules statement. For example, `['season']=S:('none');`.

Null Values in Pick Lists

All pick lists in TM1 Web (both Websheets and the Cube Viewer) always contain a null value that a user can select. The null value is automatically inserted into all pick lists in TM1 Web; it does not have to be explicitly defined.

Pick lists in TM1 Architect and TM1 Perspectives only contain a null value if the pick list is a static type and a null value has been explicitly defined for the pick list. Dimension pick lists and subset pick lists can never contain a null value when used in TM1 Architect and TM1 Perspectives.

Selecting Null Values in Pick Lists

There are two ways to select null values from pick lists.

Procedure

1. In any of the TM1 clients, you can click the pick list and then click the null value.

   Important: In TM1 Perspectives slices and Active Forms, do not click the null value in a string cell. This will delete the DBRW formula from the cell and you will no longer be able to retrieve data for that cell from the IBM Cognos TM1 server. You can safely click the null value in numeric cells.
2. In TM1 Web and TM1 Architect, you can press the **Delete** key in a cell containing a pick list to select the null value.

   Important: Do not press **Delete** to select a null value in TM1 Perspectives slices or Active Forms. This will delete the DBRW formula from the cell and you will no longer be able to retrieve data for that cell from the TM1 server.

Pick List Order of Precedence

When multiple pick lists apply to an individual cube cell, the following order of precedence is used to determine which pick list is used in the cell:

- If a pick list control cube exists and contains a pick list definition for the current cube cell, the definition in the pick list control cube is used.
- If a pick list control cube does not exist, the elements that identify the current cell are examined in reverse order in a search for Picklist element attributes. The first Picklist element attribute that is encountered in this search is used in the cell.

Replicating Cubes between Servers

Using the TM1 Replication feature, you can copy cubes and other associated objects from a remote server to your local server, or between two remote servers. You can also synchronize the data updates among the copied cubes either at specified time intervals or on demand.

Replication offers the following advantages:

- Enhances response time because you can update a cube locally without communicating across a network.
- Lets you copy the latest shared data to a laptop for presentations outside of your organization.

TM1 provides bi-directional synchronization for replicated data. During the synchronization process, TM1 checks the servers involved in a replication for the latest updates to the data, and then copies the latest updates to the other servers.

Replication creates a relationship between two cubes and between two servers. These relationships are described in "Cube Relationships" on page 33.

Cube Relationships

Replication creates a relationship between two cubes:

- *Source* **cube** - The original cube in a replication

- *Mirror* **cube** - A copy of the source cube

Depending on your access privileges, you can replicate a single cube on many different servers, and you can replicate a replicated cube.

Server Relationships

Before replicating a cube, you need to log on to a remote server and create a replication connection. Replication creates a relationship between two servers:

- *Source* **server** - The remote server you log in *to*
- *Target* **server** - The server you logged in *from*

The Server Explorer window lists the current replication connections beneath the Replications icon. In this example, regions 1 is the target server and sales is a source server.

Required Access Privileges

The following access privileges are required to replicate a cube:

- Your security group must have at least Read access to the cube you want to replicate.
- You must be the TM1 administrator on the target server. On your local server, you are always the TM1 administrator.

Chapter 3. Translating your model

IBM Cognos TM1 provides a mechanism to display objects on your TM1 server in other languages, so that users can view object names in their language without requiring any configuration.

Translation in Cognos TM1 is accomplished through the Caption attribute, which lets you assign translated names to any cube, dimension, member, or member attribute on the TM1 server. You can assign Caption attribute values for all language locales supported in TM1, which correspond to the members in the }Cultures control dimension.

When a user starts any of the TM1 clients that support translation, object names display the Caption attribute value for the language associated with the user's current locale, without requiring any configuration. If you have added translated values to the cube, translated attribute values are also displayed in the filter dialog box.

The following TM1 clients support translation:

- IBM Cognos TM1 Web
- IBM Cognos TM1 Application Web
- IBM Cognos Insight
- IBM Planning Analytics for Microsoft Excel

TM1 Web and TM1 Application Web use the current browser language setting to determine the language to display.

IBM Planning Analytics for Microsoft Excel uses the Windows Location setting to determine the language to display.

Cognos Insight uses the Windows Location setting to determine the language to display when opened from the Windows Start menu. When Cognos Insight is opened from the TM1 Application Web workflow screen, it uses the Content Language defined in the portal user preferences.

Note: IBM Cognos Performance Modeler can optionally display translated names or invariant names for objects on the TM1 server. An invariant name is the name assigned to an object upon original creation. To display translated names, right-click the root on the Model Design pane, then click **Show Captions**. To display invariant names, right-click the root on the Model Design pane, then click **Show Invariant Names**.

The Caption attribute

The Caption attribute can be set up as an Alias type or as a Text type. When the Caption attribute is an Alias type, the attribute values are used to display translated object names. Additionally, TM1 enforces the uniqueness of its Caption attribute values, and you can use the Caption value to search for the associated dimension, cube or member, or to use as arguments to functions that retrieve or send data to the TM1 server.

When the Caption attribute is a Text type, the attribute values are used solely to display translated object names. Uniqueness is not enforced, so you can use the same value for multiple attributes if desired.

Note: When defining the Caption attribute for use in TM1 Web, TM1 Application Web or IBM Planning Analytics for Microsoft Excel, define the Caption attribute as an Alias type. In TM1 Web and TM1 Application Web, cube view dimensions can be set to display only an alias. In addition, when defining a SUBNM for display in a web sheet, it can take only an alias as an argument. Similarly, in IBM Planning Analytics for Microsoft Excel, SUBNM takes an alias as an argument.

Language locale codes and behavior of the Caption attribute

TM1 uses international language codes defined by ISO 639-1 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, TM1 automatically retrieves the value of the associated major language code. For example, if a Caption attribute value does not exist for "pt-BR", TM1 retrieves the value for "pt".

If no values are found for a Caption attribute, the base default attribute value is returned.

© Copyright IBM Corp. 2007, 2017

Review the list of elements in the }Cultures control dimension to familiarize yourself with the ISO 639-1/IETF combinations supported in TM1.

Translating cube names

You display cube names in other languages by creating a TurboIntegrator process that creates the Caption attribute for all cubes on your IBM Cognos TM1 server, then assigns Caption values for the cube names you want to translate.

Before you begin

For complete details on using TurboIntegrator, see *TM1 TurboIntegrator*. For details on all TurboIntegrator functions, including CubeAttrInsert and CubeAttrPutS, see *TM1 Reference*.

About this task

You can create the Caption attribute as either an alias attribute or a string (text) attribute. The benefit of creating the Caption attribute as an alias is that an alias attribute value can be passed as an argument to other TM1 functions, while string attribute values cannot.

Procedure

1. Create a new TurboIntegrator process.
2. On the Prolog tab, create the Caption attribute.

   To create Caption as an alias attribute, enter `CubeAttrInsert( '', 'Caption', 'A');`

   To create Caption as a string attribute, enter `CubeAttrInsert( '', 'Caption', 'S');`

   This creates a cube: }LocalizedCubeAttributes dimensioned by }Cubes, }Cultures, }CubeAttributes.
3. For each cube that you want to translate, insert a CubeAttrPutS function for each language that you want to make available on your TM1 server.

   For example, if you want to display the Sales cube and Price cube in French and German, your process would 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' );
   ```
4. Save and run the TurboIntegrator process.

Results

After the process successfully completes, the TM1 clients that support translation display any translated cube names for the locale in which the client is running.

Translating dimension names

You can display dimension names in other languages by creating a TurboIntegrator process that creates the Caption attribute for all dimensions on your IBM Cognos TM1 server, then assigns Caption values for the dimension names you want to translate.

Before you begin

For complete details on using TurboIntegrator, see *TM1 TurboIntegrator*. For details on all TurboIntegrator functions, including DimensionAttrInsert and DimensionAttrPutS, see *TM1 Reference*.

About this task

You can create the Caption attribute as either an alias attribute or a string attribute. The benefit of creating the Caption attribute as an alias is that an alias attribute value can be passed as an argument to other TM1 functions, while string attribute values cannot.

Procedure

1. Create a new TurboIntegrator process.
2. On the Prolog tab, create the Caption attribute:

   To create Caption as an alias attribute, enter `DimensionAttrInsert( '', 'Caption', 'A');`

   To create Caption as a string attribute, enter `DimensionAttrInsert( '', 'Caption', 'S');`

   This creates a cube: }LocalizedDimensionAttributes dimensioned by }Dimensions, }Cultures, }DimensionAttributes
3. For each dimension that you want to translate, insert a DimensionAttrPutS function for each language that you want to make available on your TM1 server.

   For example, if you want to display the Model dimension in French and Portuguese, your process would include the following functions:

   ```
   DimensionAttrPutS( 'Modèle', 'Model', 'Caption', 'fr' );
   DimensionAttrPutS( 'Modelo', 'Model', 'Caption', 'pt' );
   ```
4. Save and run the TurboIntegrator process.

Results

After the process successfully completes, the TM1 clients that support translation display any translated dimension names for the locale in which the client is running.

Translating member names

You can display member names in other languages by creating a TurboIntegrator process that creates the Caption attribute for all members of a specific dimension on your IBM Cognos TM1 server, then assigns Caption values for the member names you want to translate.

Before you begin

For complete details on using TurboIntegrator, see *TM1 TurboIntegrator*. For details on all TurboIntegrator functions, including AttrInsert and AttrPutS, see *TM1 Reference*.

About this task

You can create the Caption attribute as either an alias attribute or a string attribute. The benefit of creating the Caption attribute as an alias is that an alias attribute value can be passed as an argument to other TM1 functions, while string attribute values cannot.

Procedure

1. Create a new TurboIntegrator process.
2. On the Prolog tab, create the Caption attribute:

   To create Caption as an alias attribute, enter `AttrInsert( '<dim_name>', '', 'Caption', 'A');`. This function creates the Caption attribute as an alias attribute for the members of the <dim_name> on your TM1 server.

   To create Caption as a string attribute, enter `AttrInsert( '<dim_name>', '', 'Caption', 'S');`
3. For each member name that you want to translate, insert an AttrPutS function for each language that you want to make available on your TM1 server.

For example, if you want to display the January member in French, German, and Portuguese, your process would 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 you add an attribute with the optional fourth parameter set for a Culture, a cube will be created: }LocalizedElementAttributes_yourDimension dimensioned by yourDimension, }Cultures, }ElementAttributes_yourDimension

4. Save and run the TurboIntegrator process.

Results

After the process successfully completes, the TM1 clients that support translation display any translated member names for the locale in which the client is running.

In TM1 Web and TM1 Application Web, you need to select to display the Caption (or other translated) alias in order for the view or SUBNM to pick up your translated values. Translated attribute values will also be displayed in the filter dialog if you have added translated values to the cube.

Chapter 4. Advanced Calculations for Business Data

This section describes how to create "drill-through" processes and rules, which enable you to link the cells with related data to provide the details or context for cube values.

For a thorough examination of rules in a business scenario, see *TM1 Rules*. This documentation contains a tutorial that steps you through developing rules in a business environment.

Note: The images of the Rules Editor in this section show the old Rules Editor. To see images of the new Rules Editor, see *TM1 Rules*.

Depending on your platform, some of the supplied example data may not be available to all users.

Overview of Cube Rules

The most common calculations in OLAP applications involve aggregating data along a dimension. In TM1, you create these calculations by using consolidation hierarchies. For example, in a Month dimension, you can define a quarterly total that sums the January, February, and March values.

In many applications, you need to perform calculations that do not involve aggregating, such as cost allocations and exchange translations. With cube rules, you can create formulas to perform these calculations.

With cube 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.

Note: You must be a member of the ADMIN group to create or edit TM1 rules.

You associate a cube rule with an individual cube. Compiled rules are stored in files that are called *cube_name*.rux. When a cube for which you define rules is loaded into memory, TM1 searches for the cube's .rux file in the data directory containing the cube. The .cub file and the associated .rux file **must** reside in the same directory or the rules will not be loaded.

When you create a rule, TM1 also generates a file called *cube_name*.blb, which contains format information for the Rules Editor.

Note: If you want to edit a .rux file in a text editor other than the Rules Editor, be sure to delete the corresponding .blb file. If you do not delete the file, there will be a discrepancy between the contents of the .rux file and the display in the Rules Editor, as the .blb file determines the display in the Rules Editor.

Guidelines for Writing TM1 Rules Statements

The general format of a rules statement is shown in the following statement.

```
[Area]=Formula;
```

Variable	Description
Area	Specifies the portion of a cube that is affected by the rule.
Formula	Describes how TM1 calculates the cells in the cube area.

© Copyright IBM Corp. 2007, 2017

To restrict a rule to simple values in the Area, use the following statement:

```
[Area]=N:>Formula;
```

To restrict a rule to consolidated values in the Area, use the following statement:

```
[Area]=C:>Formula;
```

General Considerations

- The syntax is not case-sensitive. You can use both uppercase and lowercase letters.
- You can use spaces within rules to improve clarity.
- A rules statement can occupy one or more lines in the Rules Editor. A statement can also contain one or more formulas.
- You must end each statement with a semicolon (;).
- To add comments and to exclude statements from processing, insert a number sign (#) at the beginning of a line or statement. For example:

  ```
  #
  The following rule is not active
  ```

  ```
  # ['Gross Margin']=['Sales']*0.53;
  ```

Syntax for Describing the Area

The Area identifies one or more cells in a cube.

Consider the following guidelines when you create an Area definition.

- Specify no dimension elements, or one or more dimension elements.
- Each element must be from a different dimension of the cube.
- Enclose each element in single quotes.
- Use commas to separate each element.
- Enclose the entire Area definition in brackets.

The following table shows four Area examples. Each successive example narrows the scope.

Sample Area	Scope
[ ]	All cells in the cube.
['January']	All cells identified by a January element.
['Sales','January']	All cells identified by the Sales and January elements.
['Germany','Sales','January']	All cells identified by the Germany, Sales, and January elements.

Using Subsets in an Area Definition

You can use a subset in place of a single element in an Area definition by enclosing all subset members in curly braces.

For example, the following Area definition applies a rule to all cube cells identified by the element Sales and the element January, February, or March:

```
['Sales', {'January', 'February', 'March'}] =
```

Using Special Characters and Non-unique Element Names in an Area Definition

You can use the syntax 'dimensionname':'elementname' in a rules Area definition to specify elements that are not unique to a single dimension, or for dimension names that contain special characters.

For example,

```
['Units','Mar','}Groups':'ADMIN']
```

allows you to write a rule for the }Groups dimension, which contains the curly brace (}) special character.

Similarly,

```
['Units','Mar', 'Region':'North America']
```

lets you write a rule when the element North America is not unique to the Region dimension.

Syntax for Formulas

A rules formula is an expression composed of:

- Numeric constants
- Arithmetic operators and parentheses
- Numeric and string functions -- see *TM1 Reference*.
- Conditional logic
- Cube references

Numeric Constants

The simplest components of rules formulas are numeric constants.

- Consists of numerals, an optional leading minus sign (-), and an optional decimal point. For example, 5.0, 6, -5. Some examples of invalid numeric constants are: 1-, 1A, 3..4.
- Contains a maximum length of 20 characters.
- You can use scientific notation to enter a numeric constant.

For example, the following rules statement assigns the value 200 to all cells in the cube.

```
[ ] = 200;
```

Arithmetic Operators

You can combine numeric constants with the following arithmetic operators.

Operator	**Meaning**
+ (Plus sign)	Addition
- (Minus sign)	Subtraction
* (Asterisk)	Multiplication
/ (Forward slash)	Division - returns an undefined value and displays N/A in the view when you divide by zero
\ (Backslash)	Zero Display Division - same as Division operator, but returns zero when you divide by zero.
^ (Caret)	Exponentiation

Using Conditional Logic

Use the IF function to include conditional logic in rules. The general format is:

```
IF(test, value1, value2)
```

- The IF function returns one of two values depending on the result of a logical test.
- When the expression Test is true, the IF function returns Value1.
- When the expression Test is false, the IF function returns Value2.
- The data type returned by an IF function is determined by the data types of Value1 and Value2.
- Value1 and Value2 must be the same data type, either string or numeric.
- An IF function where Value1 is a string and Value2 is a number yields an error statement.

You can also nest IF statements:

```
IF(test1, value1, IF (test2, value2, value3))
```

The following table shows two IF examples.

Expression	Result
IF (7>6,1,0)	yields 1
IF (7>6, 'True', 'False')	yields 'True'

Using Comparison Operators

You can compare numbers with the following operators.

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
=	Equal to
<>	Not equal to

To compare two string values, insert the @ symbol before the comparison operator, as shown in the following example:

```
IF ('A'@='B',0,1) yields the number 1.
```

You can combine logical expressions with logical operators.

Operator	Meaning	Example
& (Ampersand)	AND	(Value1 > 5) & (Value1 < 10) Returns TRUE if the value is greater than 5 and less than 10.
% (Percent sign)	OR	(Value1 > 10) % (Value1 < 5) Returns TRUE if the value is greater than 10 or less than 5.

Operator	Meaning	Example
~ (Tilde)	NOT	~(Value1 > 5) Equivalent to (Value1 <= 5)

You can concatenate strings using the pipe (|) character.

For example, the following expressions returns Rheingold.

```
(Rhein | gold)
```

If the string resulting from a concatenation is longer than 254 bytes, TM1 returns an error.

Using Cube References

All rules formulas contain cube references, which point to areas within a cube for data. The cube references can point to the cube for which you are writing a rule (internal cube references) or to areas within other cubes (external cube references).

Internal Cube References

Internal cube references use the same syntax as the area for which you write the rule. Examples include:

```
['January']
```

```
['Sales','January']
```

```
['Germany','Sales','January']
```

In the following example, TM1 calculates the Gross Margin for Germany by multiplying the Sales for Germany in the same cube by 0.53:

```
['Gross Margin','Germany']=['Sales']*0.53;
```

External Cube References

Use the DB function to point to external cubes.

```
DB('cube', dimension1, dimension2,...dimensionn)
```

Argument	Description
cube	Name of the external cube.
dimension	One of the following arguments: • The name of an element in a dimension of the external cube, enclosed in single quotes. • The name of a dimension preceded by an exclamation mark (!), which is called *variable notation*. An argument using variable notation returns the current dimension element in the cell to which a rule statement applies. For example, in a rules-calculated cell that is identified by the Germany element of the Region dimension, !Region returns Germany. • An expression that resolves to an element name.

Specify a dimension argument for each dimension of the external cube. You must order the dimension arguments to correspond to the order of the dimensions in the external cube.

In the following rules statement, all Sales values in the internal cube are computed by multiplying Units in the internal cube by the values in the external PriceTab cube:

```
['Sales']=['Units']*DB('PriceTab',!Region,!Product,!Month)
```

- The PriceTab cube contains only prices. Each of its cells is identified by an element in three dimensions: Region, Product, Month. The internal cube contains these dimensions and at least one other dimension that has both the Sales and Units elements.
- Every Sales cell in the internal cube is identified by Sales and elements in the three dimensions the internal cube shares with the PriceTab cube. To populate any Sales cell, TM1 pulls a PriceTab value located at the intersection of the corresponding elements in the shared dimensions.
- The external cube can differ from the internal cube in terms of the number of dimensions and the number of elements along each dimension. However, a dimension you reference as a variable (as in !Region or !Product) must at least contain all the elements found in the internal cube's corresponding dimension.

Arranging Rules Statements

When more than one statement in a rule applies to the *same* Area, the first statement takes precedence.

Consider this example. A cube named Priority has two dimensions, Region and Year. The rule has four statements:

```
['Germany', 'Year1'] = 10;['Year1'] = 5;['United States']
= 6;[ ] = 2;
```

Here are sample values for the Priority cube, all of which are derived by the preceding rule.

Region	**Year 1**	**Year 2**	**Year 3**
France	5	2	2
Germany	10	2	2
United States	5	6	6

TM1 processes the rule statements as follows:

- The first statement assigns the value 10 to the Germany, Year1 cell. The first statement takes precedence over the second statement, which specifies that all Year1 cells contain 5.
- The second statement takes precedence over the third statement. Therefore, the cell for United States, Year 1 contains 5, even though the third statement specifies that all values for United States should be 6.
- The last statement [ ] = 2 specifies that all values in the cube contain the value 2. This rule applies to all cells that are not affected by preceding statements, such as the cell France, Year2.

Specifying Different Rules at the N: and C: Levels

It is often necessary to differentiate the way C: level and N: level cells within an area are treated.

- **N: Level Cells** - Identified only by simple elements.
- **C: Level Cells** - Identified by at least one consolidated element.

Use the following syntax to write a rules statement that applies only to N: level cells in an area:

```
[Area] = N:[Formula];
```

For example:

```
['Sales'] = N:['Price']*['Units']\1000;
```

Use the following syntax to write a rules statement that applies only to C: level cells in an area.

```
[Area] = C:[Formula];
```

For example:

```
['Price'] = C:['Sales']\['Units']*1000;
```

When a specific area of a cube is calculated differently at the C: and N: levels, you can use the following syntax:

```
[Area] = N:[Formula A]; C:[Formula B];
```

For example:

```
['Price'] =
```

```
     N:DB('PriceCube', !Actvsbud, !Region, !Model, !Month);
```

```
     C:['Sales']\['Units']*1000;
```

Bypassing Rules

By using the STET function, you can bypass the effect of a rules statement for specific areas of a cube.

For example, you might want to write a rules statement for Gross Margin that applies to all regions except France. You can write the general rule and the exception in two ways.

- Write the STET statement first followed by the general statement:

  ```
  ['Gross
  Margin', 'France'] = STET;
  ```

  ```
  ['Gross Margin'] = ['Sales'] * 0.53;
  ```

- Write one rules statement that includes an IF function:

  ```
  ['Gross
  Margin'] = IF(!Region @= 'France', STET, ['Sales']
  * 0.53);
  ```

Qualifying Element Names

When you want to limit a rules statement to values identified by an element that appears in multiple dimensions, qualify the element name with its dimension name using the following syntax:

```
['dimname':'element']
```

For example, if Total occurs in both the Region and Product dimensions and you want the rule limited to cells identified by Total in the Region dimension, specify:

```
['Region':'Total']=
```

Rules Editor and Rules Worksheets

You can create rules through two interfaces:

- Rules Editor - All previous examples in this section use this interface.
- Rules worksheet - A modified Excel worksheet in which you type rules statements for one cube in column A.

TM1 saves the rules in two files:

cube **.xru** - Rules worksheet *cube* **.rux** - Compiled file

Note: You can modify the rules you create through the rules worksheets by using the Rules Editor. When you do so, TM1 prompts you to save the changes to the worksheet. If you do not save the changes, the .xru and .rux files will be out of sync. Therefore, you cannot subsequently modify the changes by editing the rules worksheet. All rules described in that section apply to the rules worksheets as well as the dimension worksheets.

To ensure that TM1 has access to all of your changes, use one of the interfaces exclusively for creating and maintaining rules.

Creating Rules Worksheets

When you use a rules worksheet to create a rule on *either* a local or remote IBM Cognos TM1 server, your TM1 client must have a valid directory set in the Local Server Data Directory box in the TM1 Options dialog box. The directory is the location where your TM1 client saves the rules worksheets (.xru files).

Note: If you do not set a directory, or if the directory is not valid, TM1 cannot save your rules worksheets.

Procedure

1. Click **TM1** > **Rule Worksheets** > **New** in Excel.

   The Select Cube for Rules dialog box opens. The list contains the cubes on your local server and any servers you have logged in to during the current TM1 session.
2. Select a cube and click **OK**.

   TM1 displays a blank rules worksheet that looks like other Excel worksheets, except the first column has a width of 100.
3. Place each rule statement on a separate line in column A, as in the following example.

   ```
   ['Gross Margin%']=['Gross Margin']\['Sales']*100;
   ```

   ```
   ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);
   ```

   ```
   C:['Sales']\['Units']*1000;
   ```

   ```
   ['Sales']=N:['Price']*['Units']\1000;
   ```

   You can click **TM1** > **Edit Formula** to access the **TM1 Edit Formula** dialog box. This dialog box, which helps you construct accurate rules statements, provides all of the functionality found in the Rules Editor. The Rules Editor is described later in this section.

Saving Rules Worksheets

To save the rule in a format that TM1 can use, click **TM1** > **Rule WorkSheets** >**Save** in Excel.

The **Save** option updates the rule worksheet file (cube.xru) and creates a compiled rules file (cube.rux). TM1 saves the .xru file in your local server data directory, and saves the .rux file in the data directory for the IBM Cognos TM1 server. TM1 immediately applies the new rules to the cube.

Note: If you click **File** > **Save** in Excel, only the cube.xru file is updated. To use the new rule in a cube, you must create the compiled rules file.

Rules and Dimension Consolidations

Rules work in concert with consolidations you define in dimensions. Although you can define consolidations using rules, this is not recommended for performance reasons. Consolidations defined in dimensions are calculated much more quickly than rules-derived values, especially in very large, sparse cubes.

Order of Calculation

Rules take precedence over consolidations within dimensions.

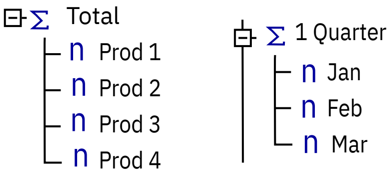
When TM1 calculates a cube cell *and* a consolidation by a rule, TM1 examines the rules statement first. However, if the rules statement refers to the cells that are the result of consolidations, TM1 first performs the consolidation and then calculates the rules statement using the results.

Conversely, if you define a cell by consolidation only, TM1 looks at the values needed to perform the consolidation. When some values are the result of calculation rules, TM1 then performs the rules calculation before performing the consolidation.

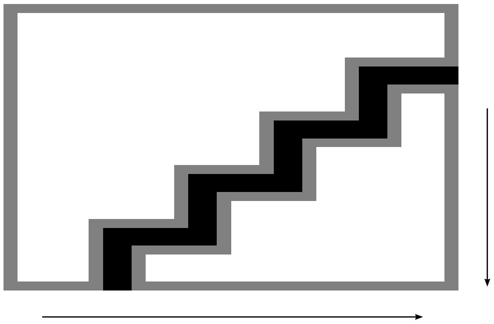
Overriding C: Level Elements with Rules

You should avoid writing a rule that overrides a consolidated value that is a component of another consolidation.

A simple example illustrates this issue. Suppose you have a two-dimensional cube named Sales that is composed of the dimensions Product and Month, with product (Total) and quarterly (1 Quarter) consolidations defined.



To calculate the grand total (Total, 1 Quarter), TM1 can consolidate the product totals for each month or consolidate quarterly totals for each product.



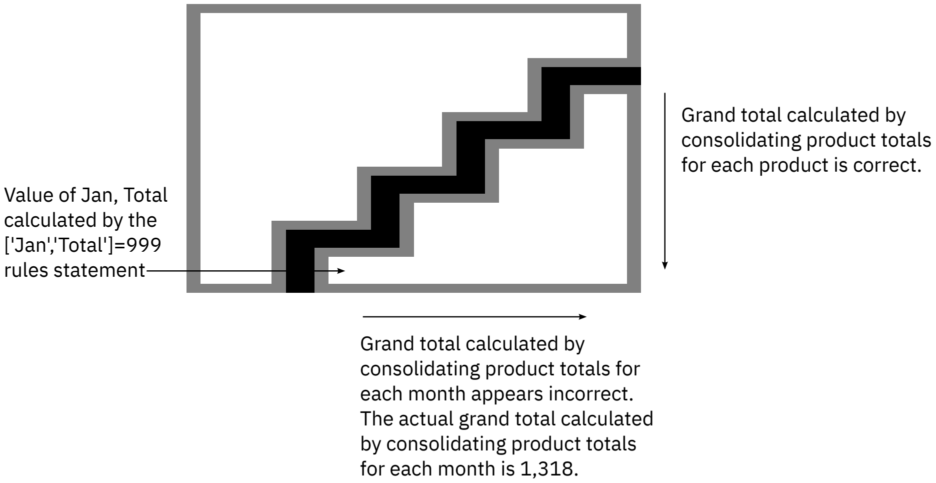
Grand total calculated by consolidating quarterly totals for each product.

Grand total calculated by consolidating product totals for each month.

Suppose further that you write a rule that calculates a value for Total product sales in Jan, and that the rules-calculated value does not sum the individual product values for Jan. A rule that defines the value of Total products in Jan as 999 serves as an illustration.

```
['Jan','Total']=999;
```

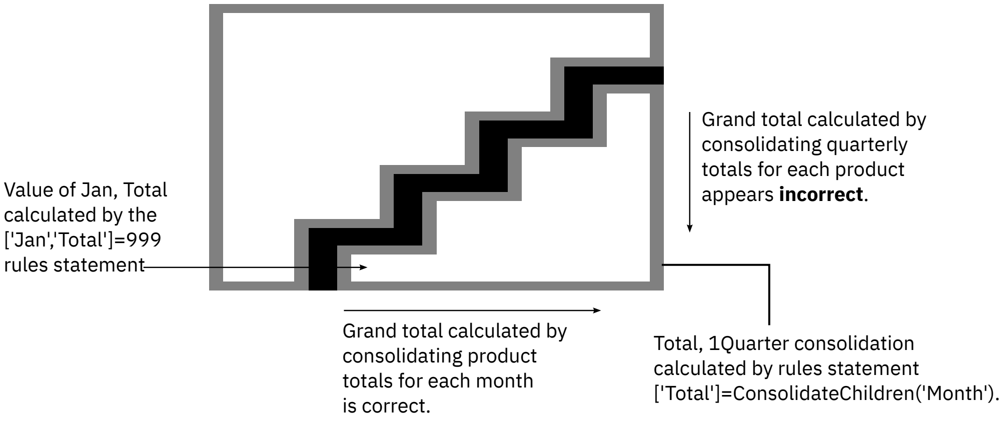
If the grand total is calculated by consolidating the product totals for each month, the value will differ from the consolidation of the quarterly totals for each product. This is because the rules-calculated value for total product sales in Jan overrides the natural consolidation defined in the Product dimension.


Value of Jan, Total calculated by the ['Jan','Total']=999 rules statement
Grand total calculated by consolidating product totals for each product is correct.
Grand total calculated by consolidating product totals for each month appears incorrect. The actual grand total calculated by consolidating product totals for each month is 1,318.

You have no control over the order in which TM1 performs dimension consolidations. Furthermore, depending on which consolidation path is optimal at any given moment, TM1 might alternate between paths. Therefore, you can request the Total, 1 Quarter value twice *in the same session*, and get different results.

You can remedy this situation by writing a rules statement that calculates the value of the Total, 1 Quarter consolidation as the sum of its immediate children along the Month dimension, thereby overriding the Product dimension consolidation. The statement ['Total']=ConsolidateChildren('Month') performs this calculation.

However, there remains an implicit inconsistency when viewing the cube: the sum of the quarterly totals for each product is different from the sum of product total for each month. Thus, overriding C: level values that are components of other consolidations is not recommended.


Value of Jan, Total calculated by the ['Jan','Total']=999 rules statement
Grand total calculated by consolidating quarterly totals for each product appears **incorrect**.
Grand total calculated by consolidating product totals for each month is correct.
Total, 1Quarter consolidation calculated by rules statement ['Total']=ConsolidateChildren('Month').

Stacking Rules

A rules statement can refer to a cell in a cube that is defined by other rules statements. TM1 stacks these rules statements until it can obtain a final value, and then works back to return a result. The number of levels of stacking that TM1 can accommodate is limited only by the available memory.

If a circular reference occurs within a rules stack, or the maximum level of stacking is exceeded, TM1 displays the error message:

```
Error Evaluating Rule: Possible Circular Reference
```

Here is an example of a circular reference:

```
['Sales'] = ['Units'] * ['Price'] ;
```

```
['Price'] = ['Sales'] / ['Units'] ;
```

Sample Applications

This section contains examples of commonly used rules applications. Examine these examples to develop an understanding of the syntax and scope of rules.

Calculating Ratios

In the following example, a rule calculates the Gross Margin as a percentage of the Sales in the SalesCube cube. You associate this ratio with the Gross Margin%, a new numeric element in the Account1 dimension.

First, you need to create the Gross Margin% element.

Creating the Margin% Element

If your SalesCube dimension already contains the GrossMargin element, skip to the next section. Follow these steps if you need to add the element GrossMargin% to the Account1 dimension.

Procedure

1. Open the Server Explorer.
2. Select the Account1 dimension.
3. Click **Dimension**, **Edit Dimension Structure**.

   The Dimension Editor opens.
4. Click **Edit**, **Insert Element**.

   The Dimension Element Insert dialog box opens.
5. Type **Gross Margin%** and click **Add**.
6. Click **OK**.
7. Save the dimension.

Creating the Margin% Rule

Follow these steps to create the Gross Margin% formula.

Procedure

1. In the Server Explorer, right-click **Sales Cube**. If the rule already exists, click **Edit Rule**. If you have not yet created the rule, click **Create Rule**.

   The Rules Editor opens.
2. Click **Area** [..].

   The Reference to Cube dialog box displays the dimensions of the SalesCube cube.
3. Click **Account1**.

   The Subset Editor window opens.
4. In the left pane, select **Gross Margin%** and click **OK**.

   The Reference to Cube dialog box reopens.
5. Click **OK**.

   The Rules Editor displays ['Gross Margin%'] in the entry field.
6. Click **Equal** -.

7. Click **Area** [..] again, and click **Account1**.
8. In the **Subset Editor** window, select **Gross Margin** and click **OK**.
9. Click **OK** in the **Reference to Cube** dialog box.
10. Click **Zero Display Division** \.

    TM1 places a division sign after ['Gross Margin'].

    Note that there are two division buttons in the Rules Editor.

    Division Operator / - If you use this division operator in a rule that results in division by zero, TM1 returns an undefined value, and displays N/A in the view.

    Zero Display Division Operator / - If you use this division operator in a rule that results in division by zero, TM1 returns the value 0.

11. Click **Area** [..] again, and click **Account1**
12. In the **Subset Editor** window, select **Sales** and click **OK**.
13. Click **OK** in the **Reference to Cube** dialog box.
14. Click **Multiplication** *.
15. Type **100** at the end of the formula.
16. Click **Semicolon** ;.

    Note: You must end all rules statements with a semicolon.

    The complete rule should now appear as follows.

    ```
    ['Gross Margin%']*['Gross Margin']\['Sales']*100;
    ```

17. Click **Save**.

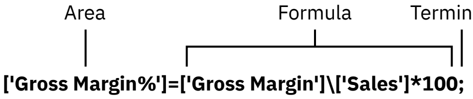
Results

TM1 saves the rule and applies it to the cube. A new Rule object opens beneath the SalesCube cube. A rule always bears the same name as the cube with which it is associated.

Understanding the Generated Rule

Let's examine the components of the new rules statement.

- **Area** - Specifies the cube area that TM1 calculates. In this example, the Gross Margin% element identifies all cell values derived through a rule.
- **Formula** - Defines the calculation.
- **Terminator** - Terminates all rules statements with a semicolon (;).



For more information about rules syntax, see *TM1 Rules*.

Browsing the Gross Margin% Values

The sample view salesmargin% now displays the calculated values for GrossMargin%.

Procedure

1. In the **Server Explorer** window, click the **Views** icon for the SalesCube cube.
2. Double-click the view **salesmargin%**.

    The view opens in the **Cube Viewer**.

The values for Gross Margin%, which are derived through the rule you just created, appear in the view.

Sharing Data Between Cubes

The SalesCube cube does not contain price data. The price information for this cube is stored in a separate four-dimensional cube called PriceCube.

You can share values between cubes by using the DB rules function. Values are stored in one cube and referenced in other cubes. The following example shows how rules for one cube can reference values in a separate cube.

Procedure

1. In the **Server Explorer**, double-click the **SalesCube** rule.

   The **Rules Editor** opens.
2. Position the cursor on the second entry line.
3. Click **Area** [..].

   The Reference to Cube dialog box displays the dimensions of the SalesCube cube.
4. Click **Account1**.

   The Subset Editor window opens.
5. In the left pane, select **Price** and click **OK**.

   The Reference to Cube dialog box reopens.
6. Click **OK**.

   The Rules Editor displays ['Price'] in the entry field.
7. Click **Equal** =.
8. Click **Database Reference** DB(..).

   The Select Cube dialog box opens.
9. Select **PriceCube** and click **OK**.

   The Reference to Cube dialog box displays the dimensions of the PriceCube cube.
10. Click **OK**.

    The following formula opens in the **Rules Editor**.

    ```
    ['Price']=DB('PriceCube',!Actvsbud,!Region,!Model,!Month)
    ```

    Read the formula as follows: Any cell in the SalesCube cube that is identified by the Price element takes its value from a cell in the PriceCube cube. The location of the PriceCube cell is found at the intersection of corresponding elements in the four dimensions that PriceCube shares with SalesCube.

    For example, the SalesCube cell identified by the elements Actual, Germany, S Series 1.8 L Sedan, Price, Jan takes its value from the PriceCube cell identified by the elements Actual, Germany, S Series 1.8 L Sedan, Jan.
11. Click **Semicolon** ; to place a semicolon at the end of the formula.
12. Click **Save** to save the rule.

    You can now change a price in the PriceCube cube and see the change reflected in the SalesCube cube. However, you cannot edit the prices in SalesCube because they are derived through the rule you just created.

    The sample view SalesPrice includes the price values.

    To open the sample view SalesPrice:
13. In the Server Explorer window, click the **Views** icon for the SalesCube cube.
14. Double-click the view **SalesMargin%**.

    The view opens in the **Cube Viewer**, complete with the Price values pulled from the PriceCube cube.

Calculating Sales

In the previous exercise, you brought prices into the SalesCube cube. If you change a price in PriceCube, TM1 does not change the corresponding *sales* value in SalesCube because the sales values in SalesCube exist as data values in the cube. You need to create a rule to *derive* the sales values in SalesCube from the prices and units.

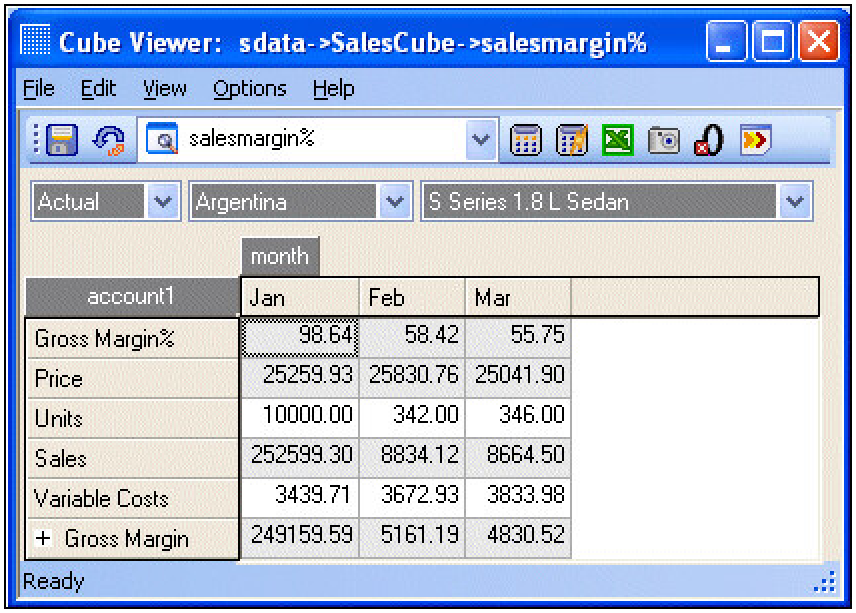
Procedure

1. Reopen the **Rules Editor** for the SalesCube cube.
2. Beneath the ['Price'] formula, enter the following rule:

   ['Sales']=['Price']*['Units']\1000;

   Note: All values in the SalesCube cube are in thousands except the Price values, which are actual numbers. Because Sales numbers should remain in thousands, you divide by 1000 in the rules statement.
3. Click **Save** to save the SalesCube rule.
4. Reopen the sample **SalesPrice** view.

   TM1 shades all cells that are identified by the Sales element, which indicates that the values in these cells are derived through rules.
5. Change the January units value to 10,000 by typing **10000** in the cell at the intersection of Units and Jan.
6. Press **F9** to recalculate the cell values.
7. Observe the new sales value for January.



account1	Jan	Feb	Mar
Gross Margin%	98.64	58.42	55.75
Price	25259.93	25830.76	25041.90
Units	10000.00	342.00	346.00
Sales	252599.30	8834.12	8664.50
Variable Costs	3439.71	3672.93	3833.98
+ Gross Margin	249159.59	5161.19	4830.52

   Note: The Gross Margin% value for January updates because this value is derived by a rule, which references the Sales element.
8. Restore the value **313** to the cell at the intersection of Jan and Sales.

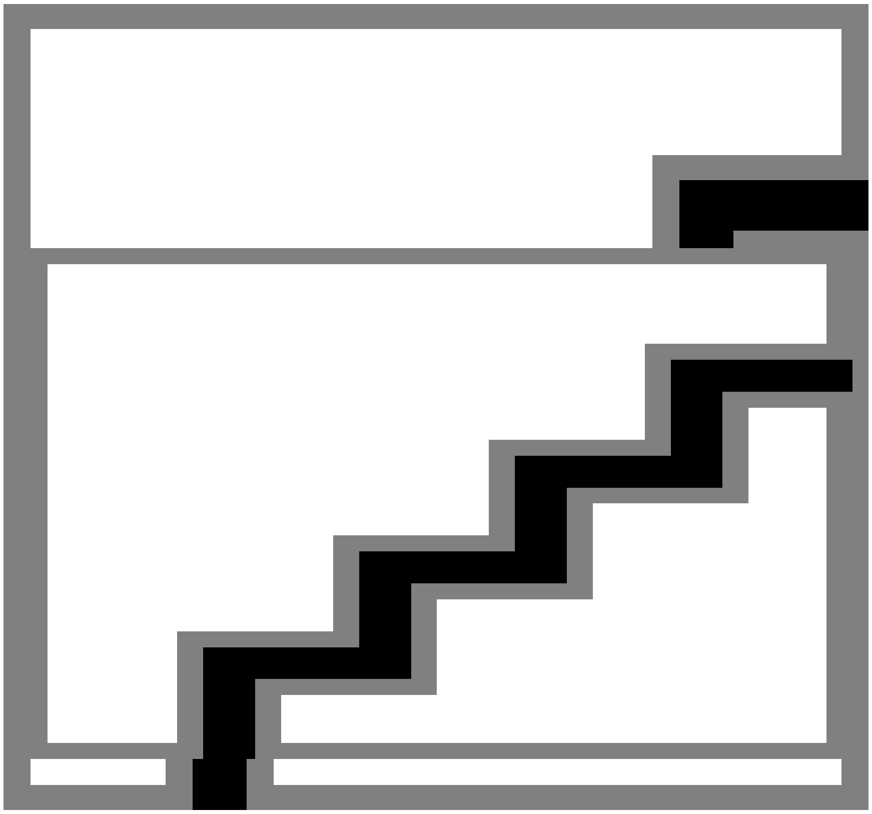
Restricting Rules to Simple Values

In the previous exercise, you created a rule that applies to all cells containing sales numbers. This type of rule supersedes consolidations within dimensions, producing incorrect results. In the following exercise, you inspect a quarterly total through a slice worksheet, and restrict the SalesCube rule to simple values, thereby allowing consolidations to function properly.

Procedure

1. Open the sample view **Sales1qtr** of the SalesCube cube.
2. Click **Slice** to slice the view to an Excel worksheet.

   The slice should appear as follows.



3. Save the slice worksheet with the name **Test**.
4. Examine the Sales value for 1Quarter in cell B10.

   According to the SalesCube rule, the value is the product of multiplying the first quarter price by a fraction (1/1000) of first quarter units. Instead, the value should be the consolidation of sales for the first three months. However, the values derived by the SalesCube rule (as it currently exists) take precedence over the values derived through consolidation. To correct this, you need to modify the rule so that it does not calculate the values for consolidated elements.
5. Open the **SalesCube** rule in the Rules Editor.
6. On the third line, insert **N:** in front of ['Price'] so that the formula reads:

   ```
   ['Sales']=N:>['Price']*['Units']\1000;
   ```

   The restrictor N: limits the rule to the cells identified only by simple elements. The rule no longer applies to consolidated elements, leaving consolidations to function properly.
7. Click **Save** to save the edited rule.
8. Press **F9** to recalculate the test worksheet.

   The correct value now opens at the intersection of Sales and 1Quarter.

Calculating an Average Price

Examine the Price, 1Quarter value in the Test worksheet. This number is the sum of the prices for January, February, and March. However, the number *should* reflect the average price for the three months. The following rules statement yields the desired value:

```
['Price']=C:['Sales']\['Units']*1000;
```

The restrictor C: limits this rules statement to consolidations; that is, only when one or more of the elements that identify a Price cell are consolidated elements.

Procedure

1. Reopen the **Rules Editor** for the SalesCube cube.
2. Enter the following statement without the Area definition, as the third line in the entry field.

   ['Price']=DB('PriceCube',!actvsbud,!region,!model,!month);C:['Sales']\['Units']*1000;

   You omit the Area definition because you already defined the Price area in the second line. When you want to apply different formulas to the same Area, you define the area and then specify the formulas sequentially.
3. Click **Save** to save the edited rule.
4. In the test worksheet, press F9 to recalculate and update the values.

   Observe the Price, 1Quarter value, which still displays the value 76,132.59. This is because TM1 uses the *first* rules formula it encounters that applies to the Price, 1 Quarter cell:

   ```
   ['Price']=DB('PriceCube',!actvsbud,!region,!model,!month);
   ```

   This statement is appropriate only for N: level cells, such as Price, Jan. Consolidated Price values should be calculated through the second Price statement. By restricting the first Price formula to N: level cells only, you enable TM1 to apply the second Price formula to consolidations.

   Note: TM1 evaluates the rules statements in the order they appear within a rule, but the first formula for a given Area takes precedence over later formulas for the same area. If you have multiple rules statements that address the same Area, you should order them least-restrictive to most-restrictive. For details, see "Arranging Rules Statements" on page 44.
5. To restrict the first Price formula to N: level cells, open the **SalesCube** rule in the **Rules Editor**.
6. Insert **N:** in front of the first formula portion of the first Price statement:

   ```
   ['Price']=N:>DB('PriceCube',!actvsbud,!region,!model,!month);
   ```

   The entire rule for the Price area should now appear as follows:

   ```
   ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);
   C:['Sales']\['Units']*1000;
   ```
7. Click **Save** to save the rule.
8. In the Test worksheet, press F9 to recalculate and display the updated values.

   All Price values should now reflect the correct calculations, with the 1 Quarter, Price value as an average of the first three months of the year.

Linking Two Asymmetrical Cubes

Most companies do not break out overhead costs by product below the Gross Margin line. These numbers, such as rent and utilities, are available only on a regional or divisional basis. In other words, the structure of the cost numbers are not symmetrical with the sales numbers. Therefore you would normally store the data in separate cubes.

The cost data that corresponds to the sales data in the SalesCube cube is available in the PnLCube cube. The two cubes compare, as follows:

- PnLCube has four dimensions. The cost data is dimensioned by version (actual versus budget), region, account, and month.
- SalesCube has five dimensions. The sales data is dimensioned by version (actual versus budget), region, product (model), account, and month.
- The two cubes share three dimensions: Region, Actvsbud, and Month.
- Measures tracked in these cubes are identified by elements in different dimensions. SalesCube uses the Account1 dimension. PnLCube uses the Account2 dimension.
- PnLCube has no data for Sales or Variable Costs. These values are already calculated in detail in SalesCube.

In the following exercise, you write rules for the PnLCube that pulls the sales and variable costs data from the SalesCube.

Procedure

1. In the **Server Explorer** window, right-click the **PnLCube**, and click **Create Rule**.

   The **Rules Editor** opens.
2. Enter two statements to specify that the Sales and Variable Costs values in the PnLCube pull the corresponding values from the SalesCube.
   - On the first line of the entry field, create this Sales formula:

     ```
     ['Sales']=DB('SalesCube', !Actvsbud,
     !Region, 'Total',
     'Sales', !Month);
     ```
   - On the second line, create this Variable Costs formula:

     ```
     ['Variable
     Costs']=DB('SalesCube', !Actvsbud, !Region,
     'Total', 'Variable Costs', !Month);
     ```
3. Click **Save** to save the rules.

   Note: Note that the Sales and Variable Costs values now appear in the cube. The Gross Margin values are also available because you defined the Gross Margin in the account2 dimension as the difference between the Sales and VariableCosts.
4. In Microsoft Excel, open the sample worksheet **TwoCubes**.

   The TwoCubes worksheet is set up to simultaneously pull numbers from SalesCube and PnLCube. This worksheet demonstrates how numbers can flow between the two cubes.

   Let's change the Units number in cell B6 to a very large number so that you can see the effect of the change ripple through the worksheet.
5. Click cell B6 and type **100000**.
6. Press F9 to recalculate the worksheet.

   Note that the Units value for the S Series 2.5 L Sedan changes the Sales value for that model and for the model total. These changes affect the Sales values of the PnLCube, which ripples all the way down to the Earnings Before Taxes value in the PnLCube.

   Note: All the cell values in this worksheet are the result of references to two cubes, SalesCube or PnLCube. The worksheet does not calculate any of the values.

Creating Drill-Through Processes and Rules

You can create a drill process and drill rules to associate a cell with more detailed data. This data can provide underlying detail for the cell, or other information relevant to the cell.

TM1 drill- through consists of two components.

- **Drill Process** - Defines the detailed data you want to associate with a cell
- **Drill Assignment Rule** - Defines the relationship between the cell and the detailed data

After you create a drill process and a drill assignment rule for a cube, you can execute the process and open the detailed data in a new window, thereby "drilling through" to a new level of detail.

Creating a Drill Process

A drill process is a TurboIntegrator process that defines the detailed data, which opens in a new window. The cube from which a drill-through originates is called the *origination cube*.

Before you create a drill process, you should be familiar with the data you want to open when drilling from the origination cube.

Procedure

1. Right-click the origination cube in the **Server Explorer**.
2. Click **Drill**, **Create Drill Process**.

   The first window of the Drill Process Setup Wizard opens.

   The wizard displays a table with parameter values for the origination cube and all its dimensions. TM1 uses these parameter values to set up the drill process. When you execute the drill process to drill from an origination cube to the detailed data, TM1 updates the parameter values to reflect the cube location from which the drill-through originates.
3. Click **Next**.

   The second window of the Drill Process Setup Wizard opens.
4. Select the **Datasource Type** for the detailed data you want to drill through to from the origination cube.

   There are three **Datasource Type** options.

Option	Description
ODBC	Drills from the origination cube to an ODBC source. The ODBC source must be accessible from the computer on which the IBM Cognos TM1 server is running. **Note:** NOTE: TM1 requires DataDirect drivers to access an Oracle ODBC source on Solaris or AIX. These drivers are not supplied with TM1 and must be acquired separately.
Cube View	Drills from the origination cube to a different cube view. You can drill to any cube that resides on the same server as the origination cube. You can define a cube view data source that exceeds the maximum amount of memory that TM1 can allocate when you access a view. By default, the memory threshold for the MaximumViewSize parameter in the Tm1s.cfg file is 100MB on a 32-bit system, and 500 MB on a 64-bit system. **Important:** If you do not specify the maximum view size in the configuration file, TM1 displays an error message when you attempt to drill to the cube view.
Other	Drills from the origination cube to any data source TurboIntegrator supports.

   TM1 displays the data source options for the data source type you selected in the wizard.
5. Define the data source.

   - For an ODBC data source, you must supply the following information.

Option	Description
Datasource Name	Name of the ODBC data source (DSN) you want to access when drilling from the origination cube.
User Name	Valid user name to log on to the ODBC source.
Password	Password for the user name.
Query	Query that defines the data to return from the ODBC source. Query results data displays in a separate window when you drill from the origination cube.

   - For a Cube View data source, you must supply the following information.

Element	Description
Datasource Name	Name of the view you want to open when drilling from the origination cube. Click Browse to select a view name or create a view.

- For the Other data source, click **Launch TurboIntegrator** to define the data source.

6. Click **Finish**.

   The Save Process As dialog box opens.
7. Enter a name for the drill process in the Name box.

   Note: For best practice, we recommend that you use a drill process name that identifies the origination cube associated with the drill process. For instance, if you create a process to drill from a cube named PriceCube to an ODBC source, you would name the drill process PriceCubeToODBCSource. This type of naming convention makes it easier to identify a drill process name when you edit a drill process, or select from several drill processes associated with a cube.
8. Click **Save**.

   TM1 saves the drill process as a TurboIntegrator process, but prefixes the name you assigned in step 7 with the string }Drill_. For example, if you save a drill process with the name PriceCubeToODBCSource, TM1 saves the process as }Drill_PriceCubeToODBCSource.

Editing Drill Processes

When you create a drill process with a Cube View data source, TurboIntegrator inserts the function ReturnViewHandle('Cube','View') above or below the **Generated Statements** area, which is located on the **Epilog** subtab of the **Advanced** tab in the **TurboIntegrator** window.

If you change the data source for a drill process, TurboIntegrator does not update the function with the new data source because the function is outside the **Generated Statements** area. You must edit the Cube View data source in the ReturnViewHandle function for the drill process.

Note: For a drill process with an ODBC data source, TurboIntegrator does not insert the ReturnViewHandle function. Therefore, you do not need to edit the function when you change an ODBC data source for a drill process.

Procedure

1. In the **Server Explorer**, right-click the origination cube with which the drill process is associated.
2. Click **Drill**, **Edit Drill Process**.

   The Select dialog box opens.
3. Select a drill process and click **OK**.

   The TurboIntegrator window opens.
4. Click the **Advanced** tab.
5. Click the **Epilog** tab.
6. Edit the **ReturnViewHandle** function to reflect the new view.

   For example, to use the Europe_1Q view of the Sales cube as a data source, the ReturnViewHandle function would look like this:

   ```
   ReturnViewHandle('Sales','Europe_1Q')
   ```
7. Click **Save**.
8. Close the **TurboIntegrator** window.

Deleting Drill Processes

Follow these steps to delete a drill process.

Procedure

1. In the Server Explorer, right-click the origination cube with which the drill process is associated.
2. Click **Drill**, **Delete Drill Processes**.

   The Delete Drill Processes dialog box opens.
3. Select the process(es) you want to delete.

- To select multiple adjacent drill processes, click and drag across the processes.
- To select multiple non-adjacent processes, hold down CTRL, and click each drill process.

4. Click **OK**.

Creating a Drill Assignment Rule

A drill assignment rule is the TM1 rule that links cube cells with related detailed data. As indicated above, the related data can be a cube view, ODBC source, or any other data source accessible through TurboIntegrator.

Procedure

1. In the Server Explorer, select the origination cube for which you want to create a drill assignment rule.
2. Click **Cube**, **Drill**, **Create Drill Assignment Rule**.

   The Rules Editor opens.

   For each cube area you want to associate with detailed data, continue with steps 3 through 8.
3. Click **Area** [..] to define the cube cells (area)you want to associate with detailed data.

   When you click the **Area** button, the **Reference to Cube** dialog box opens.
4. To define the area, do one of the following:

   - To define the area as the entire cube, click **OK**.
   - To narrow the area definition, click the dimension buttons and select the elements that define the cells you want to associate with the detailed data, and then click **OK**.
5. Click **Equal** -.
6. Click **String** S:.
7. Enter the name of the drill process enclosed in single quotation marks to define the detailed data you want to associate with the area. For example, enter 'PriceCubeToSource'.

   Caution: Do not include the }Drill_ prefix in a drill process name. For example, enter 'PriceCubeToSource' for a drill process named }Drill_PriceCubeToSource.

   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.

   Note: You can also use conditional logic or other functions to return the name of a drill process.
8. Click **Semicolon** ·.

   The semi-colon indicates the end of a rules statement.
9. Click **Save**.

Results

You can now drill through to detailed data for which you have created a drill process and a drill assignment rule.

Drill-Through Example

This section guides you through the creation of a drill process and drill assignment rule, which allows you to drill from the SalesByQuarter cube to a relational table that is the original source for the cube data. The table contains data at the monthly level, while the SalesByQuarter cube contains data at the quarterly level. By drilling through to the relational source, you can view the underlying detail for the cube data.

Setting Up the ODBC Data Source

The example in this section drills through to an ODBC source (Access database). Before looking at the example, you must set up the ODBC data source.

Procedure

1. Open the **Microsoft Windows ODBC Data Source Administrator** dialog box.

The procedure for opening this dialog box varies, depending on the version of Microsoft Windows you are running. For details, see the Microsoft Windows Help.

2. On the **System DSN** tab, click **Add**.

   The Create New Data Source dialog box opens.
3. Select **Microsoft Access Driver** and click **Finish**.

   The ODBC Access Setup dialog box opens.
4. Type **TM1 _sample_data** in the **Data Source Name** box.
5. Click **Select**.

   The Select Database dialog box opens.
6. Navigate to your \install_dir\Custom\TM1Data\PData\RelationalData directory and select **Sales.mdb**.
7. Click **OK** to exit the **Select Database** dialog box.
8. Click **OK** to exit the **ODBC Administrator** dialog box.

   The Access database named Sales is now available as an ODBC source. The example drill processes use this ODBC data source.

Creating a Drill Process

You can now create a drill process for the SalesByQuarterCube cube. The drill process defines an ODBC data source as the detailed data that you can view when you drill from the SalesByQuarterCube cube.

Procedure

1. In the Server Explorer, right-click **SalesByQuarterCube**.
2. Click **Drill**, **Create Drill Process**.

   The Drill Process Setup Wizard opens. The table contains the default parameters for the drill process.
3. Select the **Year** parameter value.
4. Click **Select Element**.

   The Subset Editor opens.
5. Select **1 Quarter** and click **OK**.
6. Repeat steps 3 through 5 to change the Gross Margin% parameter value to **Units**.
7. Click **Next**.

   The second screen of the wizard opens.
8. Select **ODBC** as the Datasource Type.
9. Click **Browse** next to the Data Source Name box and select **TM1_sample_data** .
10. Enter **admin** as the User Name.
11. Enter the following query in the **Query** box.

This query is specific to the sample Access database. It uses IIF functions that are unique to Access and cannot be used against any other database.

```
SELECT ActvsBud, Region, Model, Account1, Month, Data
FROM Sales WHERE ( ActvsBud = '?actvsbud?' AND Region = '?region?'
AND Model = '?model?' AND Account1 = '?account1?' AND
```

```
(
```

```
IIF( '?quarter?'= '1 Quarter', Month = 'Jan' OR Month
= 'Feb' OR Month = 'Mar',
```

```
IIF( '?quarter?'= '2 Quarter', Month = 'Apr' OR Month
= 'May' OR Month = 'Jun',
```

```
IIF( '?quarter?'= '3 Quarter', Month = 'Jul' OR Month
= 'Aug' OR Month = 'Sep',
```

```
IIF( '?quarter?'= '4 Quarter', Month = 'Oct' OR Month
= 'Nov' OR Month = 'Dec',
```

```
IIF( '?quarter?'= 'Year',TRUE,TRUE)))))
```

```
)
```

```
)
```

Note: Be sure to enclose the references to the TM1 parameter names in question marks (?).

12. Click **Finish**.

    The Save Process As dialog box opens.

13. Save the process as RELATIONALTABLE_SalesByMonth.

Results

The new drill process displays as an available process on the server. Note that the process name includes the **}Drill_** prefix, indicating that it is a drill process.

You can now create a drill assignment rule to make the ODBC data source available from the SalesByQuarterCube cube.

Creating a Drill Assignment Rule for SalesByMonth

Follow these steps to create a drill assignment rule that makes the RELATIONALTABLE_SalesByMonth process available from the SalesByQuarterCube cube.

Procedure

1. In the Server Explorer, right-click **SalesByQuarterCube**.
2. Click **Drill**, **Create Drill Assignment Rule**.

   The Rules Editor opens.

3. Enter the following rule in the large box.

```
['Year'] = S:IF( ( ELLEV( 'actvsbud', !actvsbud ) = 0) &
( ELLEV( 'region', !region)= 0 ) &( ELLEV( 'model', !model)= 0 ) &
( ELLEV( 'account1', !account1)  = 0),
'RELATIONALTABLE_SalesByMonth', '' );
[] = S:IF( ISLEAF= 1, 'RELATIONALTABLE_SalesByMonth','');
```

This rule indicates that the RELATIONALTABLE_SalesByMonth drill process will be executed when you select the Drill option from a cell that is either:

- Identified by the Year element and 0-level elements from all other dimensions
- Identified by all 0-level elements

4. Save the rule.

   You can now test the drill-through functionality for SalesByQuarterCube cube.

Viewing the Drill-Through Example
Follow these steps to view the result of the drill process and drill assignment rule.

Procedure

1. Open the **Drill_relational** view of the SalesByQuarterCube.
2. Right-click the cell at the intersection of Units and Year.
3. Click **Drill**.

   An extract from the TM1_sample_data data source opens, displaying the monthly detail for the cell.
4. Click **OK** to close the window.
5. Drill through other cells in the Drill_relational view to view the monthly detail. You should be able to drill through any cell in the view.
6. Change the Region title element from Germany to **Europe**.
7. Right-click any cell in the view.

Results

Note that the Drill option is not available. This is because the drill assignment rule for SalesByQuarterCube indicates that the Drill option is only available for cells identified by the 0-level elements or for the cells identified by the Year element and 0-level elements. Europe is a consolidation, so the **Drill** option is not available.

Monitoring rules statistics

You can monitor rules statistics that provide insight as to how frequently individual statements within a rule execute and how long it takes to run a rule statement.

About this task

Statistics about rule execution are stored in the }StatsByRule control cube.

Each time a rule is changed or compiled, the data for that rule is cleared and updated in the }StatsByRule control cube. This helps you to immediately see the impact of a rule change. The data in the }StatsByRule control cube does not persist between server sessions; it is cleared every time that you restart your TM1 server.

The }StatsByRule cube contains three dimension:

- }Cubes - Contains elements corresponding to each cube on your TM1 server.
- }LineNumber - Contains elements 1 through 10,000, corresponding to line numbers in a TM1 rules .rux file.

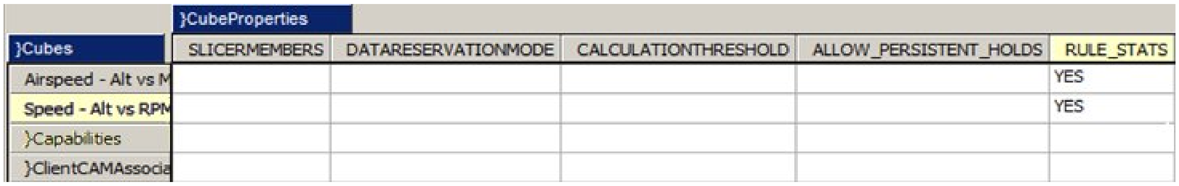
  Tip: The TM1 Rules Editor does not display line numbers. Open the .rux file in a text editor that supports line numbers to view the line numbers for a rule.
- }Rules Stats - Contains elements corresponding to the information and statistics that are collected for rules, including:
  - Rule Text - The first portion of a rule statement, provided to help you identify the statement.
  - Total Run Count - The total number of times the rule statement has run.
  - Min Time - The minimum amount of time taken for the rule statement to run, in milliseconds.
  - Max Time - The maximum amount of time taken for the rule statement to run, in milliseconds.
  - Total Time - The total amount of time taken for the rule statement to run, in milliseconds.
  - Last Run Time - The amount of time, in milliseconds, it took for the most recent execution of the rule statement.

Rule statistics collection is enabled on a per-cube basis by setting the RULE_STATS property to YES in the }CubeProperties control cube.

Note: The collection of rule statistics does incur a slight performance cost that increases with the frequency of rule execution. You should enable statistic collection only while debugging or tuning your rules. During normal operation you should disable statistic collection.

Procedure

1. Open the }CubeProperties control cube.
2. For each rule that you want to collect statistics, enter YES in the cell at the intersection of the cube name and the RULE_STATS property.



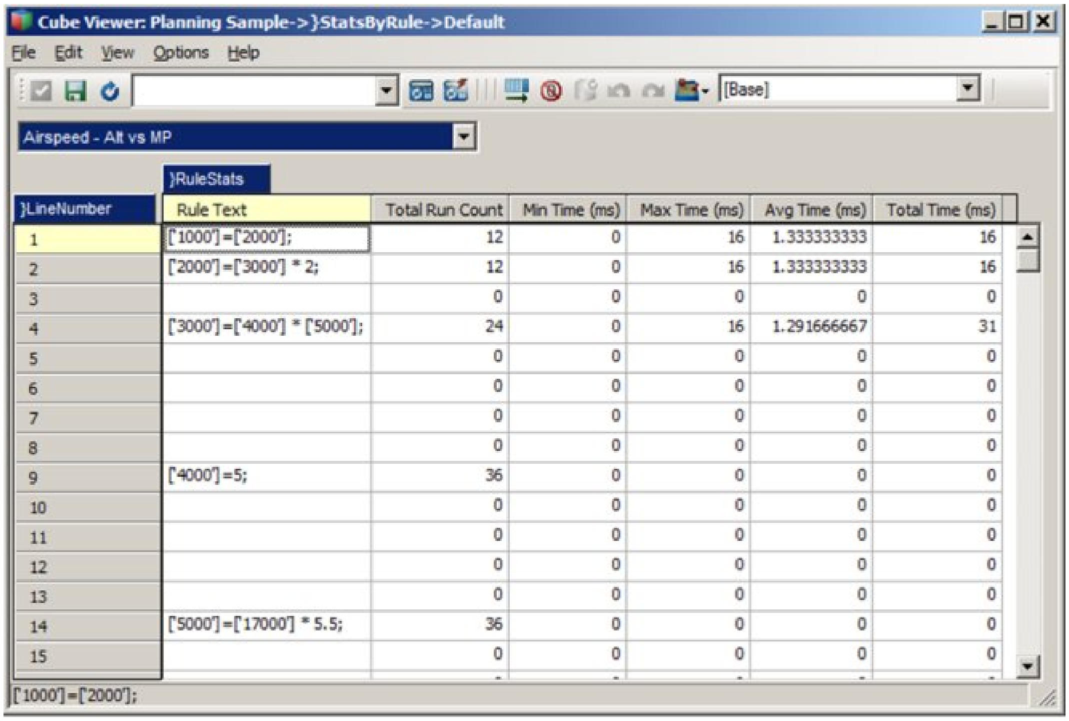
}Cubes	SLICERMEMBERS	DATARESERVATIONMODE	CALCULATIONTHRESHOLD	ALLOW_PERSISTENT_HOLDS	RULE_STATS
Airspeed - Alt vs M					YES
Speed - Alt vs RPM					YES
}Capabilities					
}ClientCAMAssocia					

Note: RULE_STATS is a dynamic parameter. It does not require a server restart to take effect, but there may be a delay of up to 60 seconds before the property is applied.

The TM1 server is now collecting statistics for the rules where the RULE_STATS property is YES. Any execution of the rules from this point forward will result in data stored in the }StatsByRule control cube.

When you want to disable the collection of rule statistics, set the RULE_STATS property to NO.

3. Open the }StatsByRule control cube.
4. Review the statistics stored for each statement in your TM1 rule. These statistics can help you identify which statements might be running more frequently than intended or are taking a long time to run. You can use this information to modify your rules.



Airspeed - Alt vs MP

}LineNumber	Rule Text	Total Run Count	Min Time (ms)	Max Time (ms)	Avg Time (ms)	Total Time (ms)
1	['1000']=['2000'];	12	0	16	1.333333333	16
2	['2000']=['3000'] * 2;	12	0	16	1.333333333	16
3		0	0	0	0	0
4	['3000']=['4000'] * ['5000'];	24	0	16	1.291666667	31
5		0	0	0	0	0
6		0	0	0	0	0
7		0	0	0	0	0
8		0	0	0	0	0
9	['4000']=5;	36	0	0	0	0
10		0	0	0	0	0
11		0	0	0	0	0
12		0	0	0	0	0
13		0	0	0	0	0
14	['5000']=['17000'] * 5.5;	36	0	0	0	0
15		0	0	0	0	0

['1000']=['2000'];

The }LineNumber elements in this view correspond to the line numbers in the associated .rux file. If a rule statement occupies multiple lines in the .rux file, the line number shown in this view is the line on which the rule statement starts.

The times recorded for Min Time, Max Time, Avg Tme, Last Run time, and Total Time are in milliseconds (one one-thousandth of a second). Some rule statements execute faster than 1 millisecond, resulting in an entry of 0 for the time. It's possible for a simple rule statement to run multiple times, while the Total Time shows as 0.

5. To view statistics for a different rule, select the associated cube name from the }Cubes dimension at the top of the view.

Debugging Rules

TM1 provides a tool called the Rules Tracer to assist in the development and debugging of rules. The Rules Tracer functionality is available only in the Cube Viewer.

With the Rules Tracer, you can do the following:

Trace calculations
Ensure that rules are being assigned to selected cells and calculated properly, or trace the path of consolidated elements

Trace feeders
Ensure that selected leaf cells are feeding other cells properly

Check feeders
Ensure that the children of a selected consolidated cell are fed properly

Tracing Calculations

To trace a calculation, whether a consolidation or a rules calculation, right-click the cell containing the calculation and click **Trace Calculation**.

The **Rules Tracer** window opens. From here, you can trace a calculation to its leaf level components. The Rules Tracer window contains two panes.

- **Tracing Calculation (top pane)** - Displays the definition of the current cell location, with an icon indicating whether the value in the cell is derived by **Consolidation** Σ or by **Rules** f(x). Also displays the current value of the cell. If the value is derived by rules, the rule displays in the status bar of the **Tracing Calculation** pane.
- **Trace (bottom pane)** - Displays the components of the first consolidated element or the first rule in the cell definition. You can double-click any item in the bottom pane to trace a path to the leaf level elements that define the cell.

Tracing a Rule Calculation Example (Simple)

Follow these steps to view a simple example of tracing a rule calculation.

Procedure

1. Open the **Trace_simple** view of the SalesCube cube.

   This view contains the Price values that are derived through rules you created earlier in this section.
2. Right-click the cell at the intersection of Price and Jan.
3. Select **Trace Calculation**.

   The **Rules Tracer** window opens.

   In the **Tracing Calculation** pane at the top, TM1 displays the current cell definition, along with **Rules** icon f(x) indicating that the cell value is derived through rules. The rule that applies to the cell is shaded, and the **Calculated** value of the cell displays in the lower right corner of the pane.

   In the **Trace** pane at the bottom, TM1 shows the evaluation of the rule that applies to the current cell. In this example, the rule

   ```
   ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);
   C:['Sales']\['Units']*1000;
   ```

evaluates to

```
['Price']=PriceCube(Actual, Germany, L Series 1.8L Sedan,
Jan)
```

because the current cell is an N: location (not defined by any consolidations).

You now know that the cell SalesCube(Actual, Germany, L Series 1.8L Sedan, Price, Jan) takes its value from PriceCube(Actual, Germany, L Series 1.8L Sedan, Jan).

Tracing a Consolidation Example (Complex)

The previous simple example shows how the Rules Tracer works for cells at the N: level. The following example shows how you can use the Rules Tracer to trace values that include consolidations.

Procedure

1. Open the **Trace_complex** view of the SalesCube cube.
2. Right-click the cell at the intersection of Sales and 1 Quarter.
3. Click **Trace Calculation**.

   The Rules Tracer opens.

   In the **Tracing Calculation** pane at the top, TM1 displays the current cell definition, along with the Consolidation icon Σ indicating that the cell value is derived through consolidation. The value of the Consolidated cell displays in the lower right corner of the pane.

   In the **Trace** pane at the bottom, TM1 displays the components of the first consolidated element in the cell definition. In this example, T Series is the first consolidated element in SalesCube(Actual, Germany, T Series, Sales, 1Quarter). TM1 displays the T Series components with their values.
4. Double-click **T Series 2.8L Coupe** in the **Trace** pane.

   You are now tracing the value of SalesCube(Actual, Germany, T Series, Sales, 1Quarter) through T Series 2.8L Coupe.

   The **Tracing Calculation** pane now displays the cell definition for the element you double-clicked.

   ```
   SalesCube(Actual, Germany, T Series 2.8L Coupe, Sales,
   1 Quarter)
   ```

   The **Trace** pane now displays the components of the first consolidated element in this cell definition. The first consolidated element in SalesCube(Actual, Germany, T Series 2.8L Coupe, Sales, 1 Quarter) is 1 Quarter. TM1 shows the three components Jan, Feb, and Mar, with their values.

   In the **Trace** pane, note that a Rules f(x) icon precedes each component name, indicating that the components derive their values through rules.
5. Double-click **Jan** in the **Trace** pane.

   You are now tracing the value of SalesCube(Actual, Germany, T Series, Sales, 1Quarter) through the T Series 2.8L Coupe and Jan.

   The Tracing Calculation now displays the cell definition for the element you double-clicked. In this case, SalesCube(Actual, Germany, T Series 2.8L Coupe, Sales, Jan). This cell is calculated by a rule, ['Sales']=N:['Price']\['Units']*1000, which is shaded. The Consolidated value of the cell, 18730.0772, displays in the lower right corner of the pane.

   The **Trace** pane now displays the components of the rule formula and their values. The component Price has a value of 43156.86, and is derived by another rule. The component Units has a value of 434, and is an Input Value, as denoted by the gray bullet ▪.

   By plugging these values into the rule formula, you can see that (43156.86 X 434)\1000 = 18730.0772, confirming the value of SalesCube(Actual, Germany, T Series 2.8L Coupe, Sales, Jan).
6. Double-click **Price** to view the rule that is used to derive the Price value.

The following rule calculates the Price:

```
['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);
C:['Sales']\['Units']*1000;
```

This rule evaluates to:

```
PriceCube(Actual, Germany, T Series 2.8L Coupe, Jan)
```

which displays in the **Trace** pane as an Input Value, denoted by a gray bullet ▣, indicating that you cannot trace the calculation any further.

Tracing a New Calculation Path

You can click any cell definition in the Tracing Calculation pane of the Rules Tracer to begin tracing a new calculation path.

For instance, using the previous complex example, you could click the first cell definition in the **Tracing Calculation** pane to begin tracing a new path through the T Series consolidation.

When you click a cell definition in the **Tracing Calculation** pane, the **Trace** pane displays the components of the first consolidated element in the cell definition. You can then double-click any component to begin tracing a new calculation path.

Tracing Feeders

The **Rules Tracer** lets you trace the way a selected cell feeds other cells.

Because you can only feed other cells from a leaf element, the **Trace Feeders** command is available for the leaf cells you define by rules, but is not available for consolidated cells.

Procedure

1. In the **Cube Viewer**, right-click the cell you want to trace.
2. Click **Trace Feeders**.

   The **Rules Tracer** window opens. This window contains two panes.

   - **Tracing Feeders (top pane)** - Displays the definition of the current cell location, and the feeder rules associated with the current cell
   - **Trace (bottom pane)** - Displays the locations fed by the current cell
3. Double-click a location in the **Trace** pane.

   This location becomes the current cell location in the Tracing Feeders pane, and the Trace pane displays any locations fed by the current cell.
4. Continue double-clicking the locations in the **Trace** pane until you have traced the feeders to the level you require.

Checking Feeders

If a cube contains a rule with SKIPCHECK and FEEDERS statements, you can use the Rules Tracer to check that TM1 properly feeds the components of the consolidation.

Procedure

1. In the **Cube Viewer**, right-click the consolidated cell you want to check.
2. Click **Check Feeders**.

   The Rules Tracer opens. This window contains two panes.

   - **Checking Feeders (top pane)** - Displays the definition of the current cell (consolidation)
   - **Trace (bottom pane)** - Displays all components of the consolidation that are not properly fed

   The Trace pane is empty, which means the consolidation is fed properly and the cubes values are accurate.

If the Trace pane displays the components of the consolidation, you must edit the rule associated with the current cube to add the FEEDERS statements that feed all the listed components.

Note: You can only check the FEEDERS for a cell once per TM1 session. The action of checking the FEEDERS actually feeds the components of the consolidation. Any subsequent checking of the FEEDERS does not yield accurate results. If you want to check the FEEDERS for a cell more than once, you must recycle the IBM Cognos TM1 server before every check.

Chapter 5. Organizing Objects in TM1 Applications

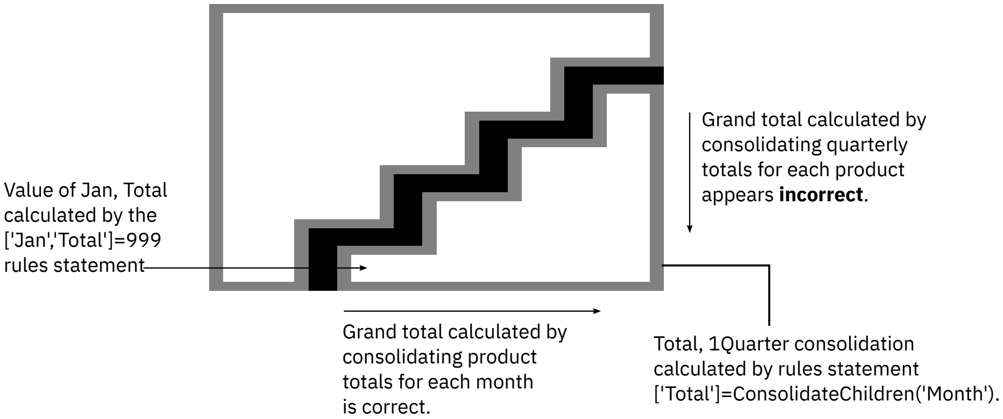
IBM Cognos TM1 lets you organize objects logically by application, and by type. This section describes how to create TM1 applications.

TM1 Application Overview

TM1 applications are objects that function as virtual folders to organize shortcuts to other TM1 objects, files, and URLs in a logical, job-specific grouping.

For example, you might create an application that organizes all TM1 objects and related files for a North American sales organization.

You insert objects, files, and URLs into an application by creating a shortcut or *reference*. Applications and references provide a quick and organized way to open the target objects to which the references point.



Value of Jan, Total calculated by the ['Jan','Total']=999 rules statement
Grand total calculated by consolidating quarterly totals for each product appears **incorrect**.
Grand total calculated by consolidating product totals for each month is correct.
Total, 1Quarter consolidation calculated by rules statement ['Total']=ConsolidateChildren('Month').

Types of References

IBM Cognos TM1 applications can contain references to any of the following items:

TM1 objects
An application can contain references to any type of TM1 object except for rules and replications.
You can reference TM1 objects on the same IBM Cognos TM1 server that contains the application or any other TM1 server to which you have access.

Files
An application can reference any type of file, such as an Excel spreadsheet, a Word document, or any other file. Any Excel file can be referenced, regardless of whether the file contains TM1 slices, functions, or other TM1 features.
You can reference both *external* and *uploaded* files.
A reference to an *external* file provides a shortcut to a file that can exist in any shared directory on your network.
A reference to an *uploaded* file saves a copy of the original file on the TM1 server. However, changes to the original source file, outside of TM1, are not automatically reflected in the uploaded copy of the file on the TM1 server. You must update the file on the server to make the changes available.

URLs
An application can contain links to any web pages or resources that use the HTTP or HTTPS protocol.

Behavior of References

It is important to understand that when you add object and file references to an application, they exist by reference only - as a shortcut that opens the original IBM Cognos TM1 object or file. Deleting a reference in an application is different from deleting the source TM1 object or file:

© Copyright IBM Corp. 2007, 2017

- Deleting a *reference* from an application has no impact on the corresponding source TM1 object or file. Only the reference is deleted.
- Deleting a *source* object in TM1, or a file on disk, breaks any corresponding reference in an application, but does not delete the reference. The reference remains but is not functional if the source object or file is deleted.

One exception to the above is for uploaded file references. In this case, the file is actually copied to the IBM Cognos TM1 server. If you delete a reference to an uploaded file, TM1 deletes the uploaded copy of the file from the TM1 server.

For details on file references, see "Adding File References to an Application" on page 73

Display Order for References in Applications

TM1 displays references in applications in the following order. Within each reference group, TM1 sorts the references alphabetically in descending order.

- Cubes
- Views
- Dimensions
- Subsets
- Processes
- Chores
- Files and URLs

Note: You cannot set the order in which reference groups display, nor the sort order within reference groups.

Using Applications and References in TM1 Web

When you create TM1 applications and references in the Server Explorer, they are automatically available in TM1 Web. This applies for references to cubes, views, files, and URLs.

For details, see "Publishing TM1 Applications to TM1 Web" on page 81.

Application Folders and Files on the TM1 server

The folders and files that support TM1 applications are stored in the IBM Cognos TM1 server data directory.

Folders

The structure for TM1 application folders is organized in the following location:

```
TM1 Data Directory \ }applications
```

Information about both referenced and uploaded files are stored in their related subfolders here.

Referenced Files

Information about referenced files are stored in placeholder files named with the `.extr` extension.

For example, if you create an application reference to a file named `sheet1.xls`, TM1 creates a placeholder file named `sheet1.xls.extr` to keep track of that entry.

Uploaded Files

Information about uploaded files are stored in placeholder files named with the `.blob` extension.

The actual files that are uploaded to the TM1 server are stored in the following location:

```
TM1 Data Directory \ }Externals
```

For example, if you upload a file to the Planning Sample database, it would be saved here:

```
C:\Program Files\Cognos\TM1\Custom\TM1Data\PlanSamp\}Externals\
```

TM1 automatically appends a date and time suffix to the end of the file name for any file you upload. The suffix uses the format *year month day time.file extension*.

For example, if you upload the file Sample_Budget.xls to the Planning Sample database, TM1 renames it to Sample_Budget.xls_20090617155650.xls and stores the file in the following location:

```
C:\Program Files\Cognos\TM1\Custom\TM1Data\PlanSamp\}Externals\Sample_Budget.
xls_20090617155650.xls
```

Security Considerations for Creating and Viewing Applications

The following table describes the security privileges required to perform actions related to TM1 applications.

For details, see "Administering Security for TM1 Applications" on page 79.

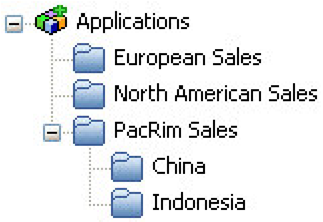
Action	Required security privilege
Create a top-level application	Must be member of ADMIN or DataAdmin group
Create a secondary-level application	Admin access to parent application
View and use applications and references	Read
Add a private reference to an application	Read
Publish a private application	Must be member of ADMIN or DataAdmin group
Publish a private reference to a public object	Admin
Publish a private reference to a private object	Cannot be done
Privatize a public application or reference	Admin
Delete a public application or reference	Admin
Delete a private application	Admin
Delete a private reference	Read

Creating and Managing Applications

To begin adding applications to an IBM Cognos TM1 server, a member of the ADMIN or DataAdmin group must first create one or more top-level applications.

Top-level applications are applications that appear directly below the Applications group in the Server Explorer. Secondary-level applications appear within a parent application, as explained in "Simultaneously Creating a New Object and Adding a Reference to an Application" on page 73.

In the following example, you see an applications group (Applications), three top-level applications (European Sales, North American Sales, and PacRim Sales), and two secondary-level applications (China and Indonesia) beneath the PacRim Sales top-level application.



Creating a Top-Level Application

Follow these steps to create a top-level application.

Procedure

1. Ensure that the Applications group is visible on your IBM Cognos TM1 server. If the group is not visible, click **View**, **Applications** in the Server Explorer.
2. In the Server Explorer, right-click the Applications group on the server where you want to create the application and click **Create New Application**.

   TM1 inserts a new top-level application temporarily named New Folder in the selected Applications group.
3. Assign a name to the new application.

   Note that Microsoft Windows has some reserved device names that cannot be used as folder or file names. See the MS Windows website for an updated list of reserved device names.

   All applications are private objects that only the user who creates the application can access. You must publish an application to make it available to other TM1 users. Only TM1 Architect and TM1 Perspectives users can create public applications. TM1 Client users can create only private applications.
4. Right-click the application and click **Security**, **Make Public** to make the application available to other users.

   Note: The type of icon that is used for an application depends on the private or public status of the application.

   - A *private* application displays with a **Private Applications** icon which includes a key in the upper right corner.
   - A *public* application displays with a **Public Applications** icon.

   You can now begin adding references and/or secondary-level applications to the application.

Renaming Applications

Follow these steps to rename an application.

Procedure

1. Right-click the application in the Server Explorer.
2. Click **Rename**.

   The current application name is selected and ready to be edited.
3. Type a new name for the application.
4. Press **ENTER**.

Deleting Applications

You can delete any application, public or private, to which you have ADMIN privilege.

Note: When you delete an application, IBM Cognos TM1 deletes all sub-applications and references contained in the application.

When you have ADMIN privilege to an application, you must also have ADMIN privilege to all references and sub-applications within the application, regardless of the security privileges that TM1 applies through the TM1 Security Assignments window. Therefore, when you have ADMIN access to an application, you can delete sub-applications and references to which you were assigned only the READ or NONE privilege.

To illustrate the ADMIN privilege for deleting an application, consider the following example. There are three applications on a server (App1, App2, and App3), all created by a member of the ADMIN group.

The administrator has defined the following security privileges to the applications for the North America user group:

Application Name	Security Privilege for North America User Group
App1	Admin
App2	Read
App3	None

When a member of the North America user group logs on to the IBM Cognos TM1 server, he will see App1, to which he has ADMIN privilege, and App2, to which he has READ privilege. He will not see App3, as he has NONE privilege for that application.

Now, if a member of the North America user group attempts to delete App1, the deletion will succeed. This is expected because the North America user group has ADMIN privilege to App1, which allows deletion of applications. However, both App2 and App3 (along with any references contained therein) will also be deleted without warning, despite the fact that the North America group has been assigned READ and NONE privileges for the applications, respectively.

Procedure

1. Right-click the application in the Server Explorer.
2. Click **Delete**.
3. Click **Yes** when prompted for confirmation.

Creating a Secondary-Level Application within an Existing Application

Any IBM Cognos TM1 user with Admin privilege to an existing application can create secondary-level applications.

Procedure

1. Right-click the existing application.
2. Click **New**, **Application**.

   TM1 inserts a new secondary-level application temporarily named New Folder in the existing application.
3. Assign a name to the new application.

   TM1 creates the application as a private object which only you can access.
4. If you are a member of the ADMIN or DataAdmin group and you want to publish the application so that other TM1 users can access it, right-click the application and click **Security**, **Make Public**.

   When you publish a secondary-level application, security privileges for the secondary-level application are inherited from the security privileges defined for the parent application. You can change security privileges by following the steps in "Assigning Security Privileges for TM1 Applications and References to User Groups" on page 79.

Adding TM1 Object References to an Application

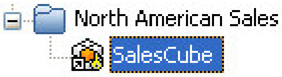
You can add references to objects from the IBM Cognos TM1 server on which the application resides as well as objects from other servers to which you are connected.

Procedure

1. Select the object in the Server Explorer.
2. Drag and drop the object onto the application.

   A reference to the object displays in the application.

By default, any reference you add to an application is a private reference, as indicated by a key superimposed over the object icon. Only you can access the reference.

Icon	Description
 North American Sales SalesCube	A reference to the SalesCube cube that was added to the North American Sales application.
	TM1 adds an image of a shortcut arrow to an object's icon to represent a reference.
	An image of a key is added to the lower corner of the reference icon to represent private references and to both lower and upper right-hand corners to represent private objects.

Making a Reference Public

If you want to make the reference available to other IBM Cognos TM1 users, you must publish the reference by right-clicking the reference and clicking **Application Item**, **Security**, **Make Public**.

TM1 combines the object's icon with a shortcut arrow to identify a *public* reference to a *public* object.

Icon	Description
	Public reference to a public subset.
	Public reference to a public view.

To simplify the process of creating a public reference, you can right-click an object in the Server Explorer, hold down the right mouse button, and then drag and drop the object onto an application. When you drop the object, TM1 displays a shortcut menu. Click **Create Public Reference**.

You can publish references in public applications only. When you publish a reference in an application, security privileges for the references are inherited from the security privileges defined for the parent application. You can change security privileges by following the steps outlined in “Assigning Security Privileges for TM1 Applications and References to User Groups” on page 79.

Adding References to Private Views and Subsets to an Application

When you add a reference to a *private* view or subset to an application, the reference is created as a private reference, which is the default behavior when creating any reference in an application.

IBM Cognos TM1 adds an image of two keys to a reference icon to identify a *private* reference to a *private* object.

Icon	Description
	Private reference to a private subset.
	Private reference to a private view.

These icons help you differentiate:

- private references to *private* objects, from
- private references to *public* objects.

TM1 adds a single key to a reference icon to identify a *private* reference to a *public* object.

Icon	**Description**
	Private reference to a public subset.
	Private reference to a public view.

If you publish a private view or subset for which a reference exists, the reference is no longer valid, and displays an error message when accessed.

For example, if create a reference to a private view called View1, and then make View1 public, the reference breaks and can not open the view. The following error message displays: Cannot find private view 'View1' of cube 'plan_BudgetPlan' on server 'planning sample'.

In such a circumstance, you should delete the old reference and insert a new reference to the (now) public view or subset.

Simultaneously Creating a New Object and Adding a Reference to an Application

You can create an IBM Cognos TM1 object from within an application. TM1 creates the object on the server where the application resides, and inserts a private reference to the object in the application. You can create dimensions, cubes, processes, and chores from within an application.

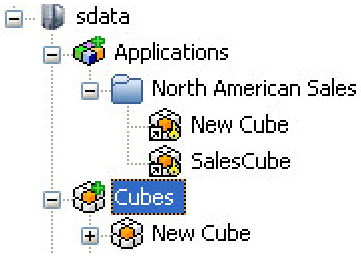
Procedure

1. In the Server Explorer, right-click the Application from which you want to create the object, and click **New <Object type>**, on the shortcut menu. For example, to create a new cube, click **New**, **Cube**.

   When you select an object type, a dialog box or window opens. For example, if you click **New**, **Cube**, the Creating Cube dialog box opens.
2. Complete the procedure required to create the type of object you chose.

   When you are done, TM1 creates the object on the server and inserts a private reference to the object in the application.

   The example shows the result of creating the cube New Cube from within the North American Sales application.



```
sdata
  Applications
    North American Sales
      New Cube
      SalesCube
  Cubes
    New Cube
```

Adding File References to an Application

You can add file references to IBM Cognos TM1 applications for any type of file on your computer or network, such as Excel, Word, PowerPoint, PDF, or any other file.

Notes:

- You can add a reference to *any* Excel file to an application. You are not restricted to working only with Excel files containing TM1 slices or other TM1 features.
- Files are opened with the program with which they are associated, as configured in the Microsoft Windows file type settings. TM1 displays an icon for each file based on this association.

Procedure

1. In the Server Explorer, right-click the Application to which you want to add the file reference, and click **Add File** on the shortcut menu.

   The Add File dialog box opens.
2. Browse to the directory containing the file and select the file you want to add to the application.
3. Select an option that determines how TM1 will add the file to the application.

   Attach the file as a reference - Keeps the file in its current location and inserts a reference to the file in the TM1 application.

   When a file is added as a reference, it displays as icon with an arrow in TM1. For example, the icon for an Excel file that is added using the Attach the file as a reference option looks like this in the TM1 application object tree.

   The primary advantage of this option is that any edits or modifications to the Excel file are immediately available in the application. If you select this option, the Excel file should reside in a shared folder and you should navigate to the file through your network to create a UNC path to the file.

   Copy the file to the TM1 Server- Copies the file to the TM1 server on which the application resides.

   This type of reference displays as an icon without an arrow in TM1. For example, an Excel file that is added using the Copy the file to the TM1 server option has the following icon in TM1 applications:

   TM1 copies the uploaded files to the }Externals subdirectory of the TM1 server data directory. For example, if you add an Excel file to an application on the sdata server that is part of the sample TM1 database, TM1 saves the file to

   C:\Program Files\Cognos\TM1\Custom\TM1Data\sdata\}Externals.

   The primary advantage to this option is that the file is available whenever the TM1 server is running. However, changes to the original source file are not automatically reflected in the file on the TM1 server. You must update the file on the server to make the changes available.

   - For details on updating Excel files that have been copied to TM1, see "Updating Excel Files on the TM1 server" on page 75.
   - For details on updating non-Excel files that have been copied to TM1, see "Updating Non-Excel Files on the TM1 server" on page 76.

   Note: If you select the **Copy the file to the TM1 server option**, the file you want to upload cannot be currently in use by another program. If you attempt to upload an open file, TM1 displays an error message. Make sure the file is closed before uploading it to TM1.
4. Select either the **Public** or **Private** option.

   - **Public** - Makes the file reference available to other users who have access to the TM1 application.
   - **Private** - Only you can access the reference to the file.
5. Click **Open**.

   The file is now available from within the TM1 application.

   You can also access files in TM1 applications from TM1 Web. For details, see the following sections later in this section:

   - "Publishing TM1 Applications and References" on page 80.
   - "Publishing TM1 Applications to TM1 Web" on page 81.

Understanding Public and Private File References

IBM Cognos TM1 uses the following icon formats to visually identify public and private file references in TM1 applications.

Icon	Description
	Public reference to a file that was added to TM1 as a reference.
	Private reference to a file that was added to TM1 as a reference.
	Public reference to a file copied to the TM1 server.
	Private reference to a file copied to the TM1 server.

Behavior of Files Uploaded to the TM1 server

An uploaded file is any file that is added to an IBM Cognos TM1 application using the **Copy the file to the TM1 server** option.

Uploaded files behave differently than referenced files because uploaded files are actually copied to and stored within the IBM Cognos TM1 server.

- TM1 copies and saves uploaded files to the TM1 server in the following directory: *<server_data_dir>*\}Externals directory.
- When a file is uploaded to the TM1 server, the file name is appended with a time/date stamp.

  For example, if you upload the file US Budget.xls to the TM1 server, the file is saved as US Budget.xls_20040702193054.xls.
- When you delete an uploaded file from a TM1 application, TM1 *deletes the copy of the uploaded file* from the }Externals directory. The original file, outside of TM1, that the uploaded file was copied from, is not deleted.
- If you want to create a hyperlink that opens an uploaded file from a Websheet or other TM1 application file in TM1 Web, the hyperlink must include the TM1 assigned name for the uploaded file. For details, see "Creating Hyperlinks to Uploaded Files" on page 82.

Updating Excel Files on the TM1 server

IBM Cognos TM1 provides a dedicated process to just update Excel files that have been uploaded to the IBM Cognos TM1 server. For details on updating *non*-Excel files (Word, PowerPoint, or other file types) see "Updating Non-Excel Files on the TM1 server" on page 76.

Procedure

1. Double-click the file in the Server Explorer.

   The file opens in Excel with a temporary file name such as TM12C5D.xls displayed in the title bar.

   Note: It is important to remember the original name of the file you are updating (as opposed to the temporary file name). You will need to select the original file later in step 4.
2. Apply your edits to the Excel document.
3. Click **TM1** > **Save Workbook on TM1 Server** > **Update Existing Application File on TM1 Server**.

   The Select TM1 External File to Update dialog box opens.
4. Select the original Excel file you want to update.

Be sure to select the original Excel file you opened in Step 1. If you select any other file, TM1 will overwrite that selected file without warning.

5. Click **OK**.

   TM1 updates the file on the TM1 server. The Excel file is available from within its parent application.

Updating Non-Excel Files on the TM1 server

You can update a non-Excel file that has been uploaded to the IBM Cognos TM1 server by saving the file as an external file and then manually re-adding the file to the server.

Procedure

1. Double-click the file in the Server Explorer.

   The file opens in the associated program with a temporary file name such as TM163.doc displayed in the title bar.

   For example, an uploaded Word file opens in Microsoft Word.
2. Using the program associated with the file, make your changes to the file and then use the program's **Save As** feature to save the file to a new location and filename.

   Note: Remember the name and location of the new updated file so you can add it back into TM1 in the next step.
3. In Server Explorer, right-click on the old version of the file and click **Delete**.

   TM1 displays the **Confirm Delete** dialog so you can delete the old file.
4. Re-add the updated file using the steps described in "Adding File References to an Application" on page 73.

Adding URL References to an Application

You can add a URL address to an application for the http:// and https:// protocols.

When you open a URL reference in an IBM Cognos TM1 application from within TM1 Server Explorer, or TM1 Web, the target source of the URL displays in your system's default web browser.

Procedure

1. In the Server Explorer, right-click the Application to which you want to add the URL reference, and click **Add URL** on the shortcut menu.

   The Add URL dialog opens.
2. Enter a complete URL, including the http:// or https:// protocol.

   For example: http://www.Company.com
3. Enter a descriptive name for the URL.

   For example: Company Web Site
4. Click **OK** to add the URL.

   The URL is added to the TM1 application, using your system's default icon for a URL link.

Working with Object, File, and URL References in TM1 Applications

You can double-click a reference to an object, file, or URL in an IBM Cognos TM1 application to perform the default action on the object.

You can right-click an object reference, and select any supported action for the object from the shortcut menu.

The following table describes the default action for all objects, files, and URLs that you can access from TM1 applications.

Reference Type	Default Action
Cube	Opens the default view of the cube in the Cube Viewer.
Cube view	Opens the view in the Cube Viewer.
Dimension	Opens the default subset of the dimension in the Subset Editor. If a default subset is not defined, the All subset is opened.
Subset	Opens the subset in the Subset Editor.
Process	Opens the process for editing in the TurboIntegrator window.
Chore	If the chore is inactive, opens the chore in the Chore Setup Wizard. (You cannot open an active chore.)
File	Opens the file in the program with which it is associated, as configured in the Microsoft Windows file type settings. For example, an .xls file opens in Excel.
URL	Opens the URL in your system's default web browser.

Note: When accessing TM1 objects that are located on another IBM Cognos TM1 server, the server must be running and you must be logged in to it.

- If you try to access a reference to an object that resides on a server that is running, but to which you are not currently connected, TM1 prompts you to log in to the server.
- If you try to access a reference to an object that resides on a server that is not running, TM1 issues the following warning: The server on which this object resides is not responding. Refresh Server Explorer display?

To restore access to the reference, start the server on which the source object resides.

Viewing TM1 Application Properties

Follow these steps to view the properties of references and sub-applications in an IBM Cognos TM1 application.

Procedure

1. Select the application in the Server Explorer.
2. If the **Properties** pane is not visible in the Server Explorer, click **Display**, **Properties Window**.

   The **Properties** pane displays information about all references and sub-applications to which you have at least Read access. References and sub-applications to which you have None access do not appear in the Server Explorer, so you cannot view the properties of these items.

   Only immediate sub-applications of the selected TM1 application appear in the Properties pane.
3. To sort items in the **Properties** pane alphabetically by property value, click the column label to which you want to apply the sort. For example, to sort items alphabetically by current status, click the **Status** column label.

   For each reference and sub-application, the following properties display.

Property	Description
Name	The name of the reference or sub-application as it appears in the selected application. You can edit reference names, which do not have to directly correspond to source object names.

Property	Description
System Name	• For most references, System Name is the actual name of the source object to which a reference points. • For files that have been uploaded to the IBM Cognos TM1 server, System Name is the name assigned to the file on the TM1 server. Naming conventions for files uploaded to the TM1 server are described in "Behavior of Files Uploaded to the TM1 server" on page 75. • For files that are attached as a reference, System Name is the UNC path to the file. • The System Name property does not apply to sub-applications.
Server	The TM1 server on which the source object for a reference resides.
Private	This property applies only to subsets and views. The Private property indicates whether the source for a subset or view reference is a private object. A property value of Yes indicates that the source is a private object. No indicates a public object.
Status	This property indicates the current availability of references and sub-applications. There are three possible Status values: • **Available** - The reference or sub-application is available for use. • **Not Connected** - You are not connected to the TM1 server on which the source object for the reference resides. Log in to the server to restore your access to the reference. • **Not Available** - The TM1 server on which the source object for the reference resides is not running, and therefore you cannot access the reference.
Security	This property indicates your security privilege for a reference or sub-application.

Deleting Object, File, and URL References from TM1 Applications

Follow these steps to delete an object reference from an IBM Cognos TM1 application.

Procedure

1. Right-click the object reference in the application.

   Note: You must select the object reference in the application. If you select the source object elsewhere in the server hierarchy, you cannot delete the object from the application.
2. Click **Application Item**, **Delete**.

   To delete a file or URL reference from an application:

   - Right-click the file in the application.
   - Click **Delete**.

Renaming Object, File, and URL References in TM1 Applications

A reference to an object or file in a IBM Cognos TM1 application does not have to use the name of the source file with which it is associated. You can rename an object or file reference in an application and maintain the connection to the source file.

Procedure

1. Right-click the reference in the application.
2. Select the **Rename** option as follows:

   - Click **Application Item**, **Rename** to rename an object reference.
   - Click **Rename** to rename a file or URL reference.

The reference name is selected and ready to be edited.

3. Type a new name for the reference.
4. Press Enter.

Deleting Source Objects that are Referenced by TM1 Applications

When you delete a source object that is referenced by an IBM Cognos TM1 application, TM1 does not delete the corresponding object reference from the application.

For example, if you delete the Canada Sales view from the data server, the reference to the Canada Sales view remains in the North American Sales application.

If you attempt to open an object or file reference in an application, and the source for the object or file has been deleted from the IBM Cognos TM1 server, TM1 displays an error message, stating that the object 'US Sales' on server 'sdata' cannot be found and prompts you to delete the reference.

When the source for an object or file in an application has been deleted from the server, you should delete the corresponding reference from the application. For details, see "Deleting Object, File, and URL References from TM1 Applications" on page 78.

Administering Security for TM1 Applications

The following sections describe how to assign security privileges for IBM Cognos TM1 applications and references to user groups on the IBM Cognos TM1 server, as well as how to publish and privatize TM1 applications and references.

Assigning Security Privileges for TM1 Applications and References to User Groups

You can assign security privileges for public items (either references or sub-applications) within public IBM Cognos TM1 applications to user groups on the IBM Cognos TM1 server. You must have Admin privilege to an application to assign security to items within the application.

You cannot assign security for private applications and references; only the user who creates a private item can access the item.

Procedure

1. In the Server Explorer, right-click the TM1 application that contains the items to which you want to assign security.
2. Click **Security**, **Security Assignments**.

   The Security Assignments window opens. The window lists all public items (TM1 objects, files, URLs, and sub-applications) that reside in the current application.
3. Select the cell at the intersection of the item for which you want to define security and the user group to which you want to assign security.
4. Click one of the available security privileges.

Security Privilege	As applied to TM1 applications	As applied to references
None	Members of the user group cannot see the application or its contents.	Members of the user group cannot see the reference.
Read	Members of the user group can see the application and use any references within the application to which the group has at least Read privilege. Members can also create private references in the application.	Members of the user group can use the reference.

Security Privilege	As applied to TM1 applications	As applied to references
Admin	Members of the user group can see the application, use references within the application, and create both public and private references in the application. They can also create private sub-applications. Members with Admin privilege to an application can set security privileges for all references and sub-applications within the application.	Members of the user group can use the reference. They can also update or delete the reference. They can publish private references, and privatize public references.

5. Repeat steps 3 and 4 for any other items for which you want to define security.
6. Click **OK**.

Results

For any given user group, it is possible to assign the READ or ADMIN privilege to a reference when the privilege assigned to the object associated with the reference is NONE. In this scenario, members of the user group will not be able to see the reference in an application.

For example, if a user group is assigned NONE privilege for a cube, but READ privilege for a reference to the same cube, members of the user group will not see the reference to the cube in an application.

Publishing TM1 Applications and References

The following sections describe how to publish IBM Cognos TM1 applications and references. To determine the security privileges required to perform these procedures, see "Security Considerations for Creating and Viewing Applications".

Publishing Private TM1 applications

Follow these steps to publish a private application.

Procedure

1. Right-click the application in the Server Explorer.
2. Click **Security**, **Make Public**.

   Note: When you publish a private application, IBM Cognos TM1 also publishes all private references to public objects within the application.

Publishing Private References to Public Objects

You can publish private references that reside in public applications

Procedure

1. Right-click the reference in the Server Explorer.
2. Select **Application Item**, **Security**, **Make Public**.

Privatizing TM1 Applications and References

You can privatize public IBM Cognos TM1 applications and references. When you make an application or reference private, only you can access the item.

Privatizing a Public TM1 Application

Follow these steps to privatize a public application.

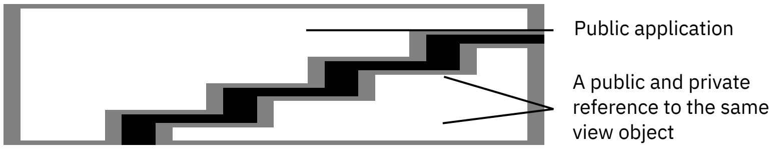
Procedure

1. Right-click the application in the Server Explorer.
2. Select **Security**, **Make Private**.

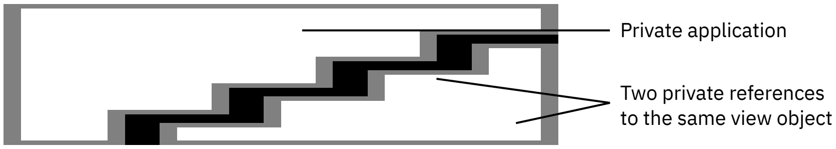
   When you privatize a public application, all public references within the application are automatically privatized as well.

If a public application contains identically named references to a single object type, one public and one private, the string _Public is appended to the public reference when the application is privatized.

For example, the following image shows the European Sales application, which contains two references to views named Northern Europe Sales, one public and one private.



When you privatize the European Sales application, the public reference is converted to a private reference and its name is changed to Northern Europe Sales_Public, indicating that this is a private reference to a public object.



The name change is necessary because an application cannot contain two identically named private references to a single object type.

Privatizing a Public Reference

Follow these steps to privatize a public reference.

Procedure

1. Right-click the reference in the Server Explorer.
2. Select **Application Item**, **Security**, **Make Private**.

Viewing Logical Groupings in TM1 Applications

One of the primary advantages of IBM Cognos TM1 applications is that they let you view and manage objects and files in logical groupings. This simplifies the task of using TM!, because you can easily identify and locate files and objects in job-specific applications without having to scan through large lists of objects arranged by type.

To further simplify the use of TM1, you can suppress the display of objects by type. This yields a cleaner, more easily navigated display in the Server Explorer. (By default, all object types are displayed in the Server Explorer.)

To suppress the display of a particular type of object, click **View**, **<Object type>** from the Server Explorer. This clears the check mark next to the object type in the View menu and suppresses the display of the object type in the Server Explorer. In the following sample, all objects are suppressed except for Applications.

Note: When you suppress the display of a given object type in the Server Explorer, references to objects of that type still appear within TM1 applications. However, control objects are suppressed within applications when the display of control objects is suppressed in the Server Explorer.

Publishing TM1 Applications to TM1 Web

All references to cubes, views, files, and URLs in IBM Cognos TM1 applications are automatically available in TM1 Web.

As of IBM Cognos TM1 version 10.2, to view a Microsoft Excel worksheet as a Websheet in Cognos TM1 Web, the file must be in the `.xlsx` or `.xlsm` format for Microsoft Excel 2007 or later.

Assume North American Sales is a public application that contains references to a collection of TM1 objects and Excel files. When you access the sdata IBM Cognos TM1 server through TM1 Web, these references are displayed under Applications.

When working in TM1 Web, you can click on a reference to open and display it as follows:

- Cubes and views display directly in TM1 Web.
- Excel files display as TM1 Websheets directly in TM1 Web.

  Note: Excel files that have been protected through the Excel command, **Tools**, **Protection**, cannot be accessed through TM1 Web.
- Non-Excel files open and display in their associated program.

  Note: Some file types and programs may not be viewable from TM1 Web.
- URL references open and display in a separate web browser.

For example, click on an Excel file to open it in TM1 Websheet format.

Note: TM1 determines the column widths of the Websheet based on the Excel file from which the Websheet is generated. If the columns in the Excel file do not accommodate the full display of row and column labels, the corresponding labels in the Websheet are truncated.

You can access both public and private applications and references through TM1 Web. Only the user who creates a private application or reference can access the item through TM1 Web. Access to public applications and references in TM1 Web is determined by the security privileges defined for the source applications and references on the TM1 server. For details on setting security privileges, see "Assigning Security Privileges for TM1 Applications and References to User Groups" on page 79.

Setting TM1 Websheet Properties

Websheet properties are configured in Server Explorer to control how an Excel file appears and behaves when viewed as a Websheet in IBM Cognos TM1 Web.

Procedure

1. In the Server Explorer, open the TM1 application containing the Excel file from which the TM1 Websheet is generated.
2. Right-click the Excel file.
3. Click **Properties**.

   The TM1 Web Properties dialog box opens.
4. Use the options on the **General** and **Display Properties** tabs to set properties for the TM1 Websheet.

   For details on the TM1 Web Properties options, see *TM1 Perspectives, TM1 Architect, and TM1 Web*.
5. Click **OK**.

Creating Hyperlinks to Uploaded Files

If you want a Websheet to contain a hyperlink to an uploaded file, the hyperlink must include the location and name that IBM Cognos TM1 assigns to the uploaded file.

When you add an uploaded file to a TM1 application, a copy of the file is saved on the IBM Cognos TM1 server and the file name is appended with a data and time stamp. For example:

```
Report_2006.xls_20070123212746.xls
```

If you do not include the TM1 assigned file name in the hyperlink, the link does not work in TM1 Web and an error displays, for example,

"File does not exist: TM1://planning sample/blob/PUBLIC/.\}Externals\upload_test_2.xls".

Procedure

1. In Server Explorer, use the **Properties** pane to find the **System Name**, which is TM1 assigned name for the uploaded Excel file that will be the target of the hyperlink.

2. Create the hyperlink to the uploaded Excel file using the following format:

   ```
   TM1://ServerName/blob/PUBLIC/.\}Externals\
   Filename
   ```

   where:

   - *ServerName* is the TM1 sever name where the Excel file is located.
   - *Filename* is the name that TM1 assigned to the uploaded Excel file.

   For example:

   ```
   TM1://sdata/blob/PUBLIC/.\}Externals\Report_2006.xls_20070123212746.xls
   ```

3. In Excel, add the hyperlink to the worksheet where you want the link to exist.
4. Add the worksheet to a TM1 application and then view the file as a Websheet in TM1 Web.

Viewing Websheets that Contain the 0x1A Hexadecimal Character

IBM Cognos TM1 Web cannot open a Websheet that contains the 0x1A hexadecimal character. If you attempt to open a Websheet containing the 0x1A hexadecimal character, TM1 Web issues the following error:

Error occurred while converting the MS Excel workbook into XML format : '', hexadecimal value 0x1A, is an invalid character. Line 54, position 34.

If you remove the 0x1A hexadecimal character from the Websheet, the file will open in TM1 Web.

Note: The ASCIIOutput TurboIntegrator function places the 0x1A hexadecimal character at the end of all generated files. If you use ASCIIOutput to export TM1 data to an ASCII file and then attempt to open the file in the TM1 Websheet, you will encounter this error.

Chapter 6. Importing Data with Processing Worksheets

This section describes how to import data into an IBM Cognos TM1 cube using a processing worksheet. A processing worksheet is a modified Excel worksheet in which you use TM1 functions to send values to a location in an existing cube.

Note: Processing worksheets, while a valid means of importing data, are deprecated functionality in the current TM1 release. We strongly recommend using TurboIntegrator to import data into TM1 cubes.

The following topics are described in this section.

- Processing Worksheets Overview
- Importing Data Using Processing Worksheets

Processing Worksheets Overview

A processing worksheet is a modified Excel worksheet in which you use functions to send input values to a location in an existing TM1 cube.

You can use processing worksheets to convert input values that do not map directly to existing elements. For example, in the following table, the first two columns contain codes that do not directly correspond to elements in the cube to which values are being sent.

Scenario	Region	Model	Measure	Jan	Feb	Mar
001	R54	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
002	R54	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
001	R32	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
002	R32	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
001	R1A	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
002	R1A	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
001	R30	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90
002	R30	S Series 1.8 L ...	Price	25259.93	25830.76	25041.90

In this example, the codes in the first column represent elements in the Actvsbud dimension. By using a simple IF function, you can convert 001 to Actual and 002 to Budget.

The second column contains four region codes, which map to regions such as Argentina, United States, and Greece. If these were the only values, you could write a nested IF formula to convert the values to elements. As the number of possible values increases, writing a nested IF formula can be cumbersome and error-prone. As an alternative, you can create a two-dimensional cube that serves as a lookup table for retrieving element names.

Importing Data Using Processing Worksheets

You can create a processing worksheet to perform the following tasks:

- Import data from input rows that require transformations.
- Update cubes but not create cubes, nor create consolidations.

© Copyright IBM Corp. 2007, 2017

- Use as a staging area for importing data.

TM! reads the input records, one at a time, into the first row of the processing worksheet, and then sends the data values associated with the record to a TM1 cube.

Beneath the first worksheet row, a processing worksheet includes:

- Conversion instructions for values that map to element names but do not match element name spellings.
- Any data transformation calculations that modify data values before importing.
- Database Send (DBS) formulas that map the input data to cells in the cube. Each formula sends a value from the first row to a location in the cube identified by one element in each dimension of the cube.

  Note: You must use DBS formulas, not DBSW formulas, in processing worksheets. You must use DBR formulas rather than DBRW formulas in processing worksheets.
- Other values in each input row supply the element names directly or through conversion instructions.

The following process summarizes the steps required to import data:

- Read in the first input record into the processing worksheet as an example.
- Compare the input with the cube structure.
- Map the input values to element names, if necessary.
- Build a DBS formula for each input value that populates a cube cell.
- Process all input records.

Reading in the First Input Row

You can use processing worksheets to process data from the following data sources:

- ASCII files
- ODBC data sources
- TM1 cubes

The next three sections provide the steps for reading an initial record from each data source.

Procedure

1. Create a new Excel worksheet, and close any other ones.
2. Click **TM1** > **Process Data** > **Example**.

   The Select Cube, ODBC or Flat File dialog box opens.
3. To choose a data source, click **Cube**, **ODBC**, or **Flat File**.

   The selection dialog box opens.
4. Select the source cube, data source or input file and click **OK**.

   For example, for an input file, select price.cma file in your \install_dir\PData directory. For an ODBC data source, change the Client and Password, if necessary.

   For an input file, the first record of the input file displays in the first row of the processing worksheet.
5. For an input file, click **Edit**, **Save** and save the processing worksheet as PriceProcessing.xls.
6. For an ODBC data source, select a table and click **OK**.

   The first record of the ODBC source displays in the first row of the worksheet.
7. For a source cube, click **Export**.

   The first record of the source cube displays in the first row of the worksheet.

Comparing Input Records with a Cube's Structures

The examples in this manual process data into the sample SalesCube cube, which has the following structure.

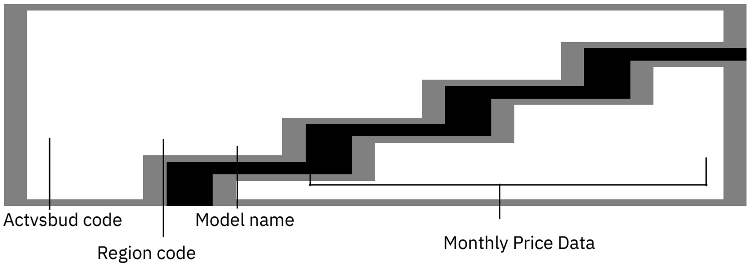
Dimension	Sample Elements
Actvsbud	Actual, Budget
Region	Argentina, Belgium, United States
Model	S Series 1.8 L Sedan, S Series 2.0 L Sedan
Account1	Units, Sales, Price
Month	Jan, Feb, Mar, Apr

To populate the SalesCube cube, each record in the data source must contain the following detail:

- One or more cell values.
- Names of elements from different dimensions that identify the cell location for each imported value, or input values that you can map to element names.

The element information can be incomplete. For example, when the source records contain multiple values for a single measure, the measure is optional. You can supply the missing measure using a DBS formula.

Consider the following record, as it appears in a processing worksheet. This record contains monthly price data for a single car model. Mapping this data to the SalesCube cube first requires converting two input values to element names and supplying missing element names.



Let's examine each input value:

- Cell A1 contains a code that identifies the price data as actual or budget amounts. Code 001 represents Actual, and Code 002 represents Budget, which are two elements in the Actvsbud dimension. By using the Excel IF function, you can convert these values to element names.
- Cell B1 supplies a region code that corresponds to an element name in the Region dimension. Suppose that the input has 21 region codes that require conversion. To convert these values, you can create a two-dimensional cube that serves as a lookup table.
- Cell C1 supplies car model names, exactly as found in the Model dimension. No conversion is required.
- Cells D1 through I1 supply the monthly data. You need to map this data to elements in the Month dimension.

Missing from each source record is a cell containing Price, which is an element in the Account1 dimension. You specify this value directly in the DBS formulas that send the data values to the cube.

Converting Using IF Formulas

The first column in price.cma contains a scenario code, 001 for actual and 002 for budget. You can use the Excel IF function to convert the code to the name of the corresponding element from the Actvsbud dimension.

Procedure

1. Click cell A3 of the processing worksheet.
2. Enter the following formula:

```
=IF(A1="001","Actual","Budget")
```

Mapping Using Fixed Labels

Cells D1 through O1 contain values that map to the 12 elements in the Month dimension (Jan - Dec). Because these input columns always map to the same months, you can enter the element names directly in the processing worksheet.

Element names must exactly match the spellings in the dimension. You can avoid misspelling names by copying them from the Subset Editor window.

The following steps illustrate how to copy element names from the Subset Editor window.

Procedure

1. Open the Server Explorer.
2. Double-click the Month dimension.

   The Subset Editor window opens.
3. Select the twelve months Jan - Dec in the **Tree** pane.
4. Click **Edit**, **Pick Elements**, **Horizontal**.
5. Return to the processing worksheet.
6. Right-click cell **D3** and click **Paste**.

   TM1 pastes the element names horizontally starting in cell D3.

Converting Using a Lookup Cube

Column B, the second input column, supplies the codes that identify the 21 regions in which the car models are sold. For example, R54 represents Argentina. To convert these codes to element names, you have two choices:

- Create a nested IF formula. As the list of codes increases, this becomes a cumbersome choice.
- Create a two-dimensional cube that serves as a lookup cube for the region names, and then retrieve the names using a DBR formula.

We'll create a lookup cube called Translate that contains two dimensions, RegCodes and RegName.

Importing Unique Names

Using TurboIntegrator, you can create a dimension whose elements are unique values from an input column. In this example, the second column is price.cma.

Procedure

1. Open the Server Explorer.
2. In the **Tree** pane, right-click **Processes** and click **Create New Process**.

   The TurboIntegrator dialog box opens.
3. Specify an **ASCII** data source type.
4. Click the Data Source Name **Browse** button and browse to the price.cma file in your \install_dir\Pdata directory.
5. Click the **Variables** tab.
6. Specify a **Content** type of **Ignore** for all columns except the one that supplies the codes you want to import. In this example, column 2 (identified by a sample value of R54) supplies the codes you want to import.
7. Click the **Maps** tab.
8. Specify **No Action** in both the **Cube Action** and **Data Action** sections of the Cubes subtab.
9. Click the **Dimensions** subtab, and do the following:

   - Type **Translate** in the **Dimension** field.
   - Select **Create** from the **Action** list.
   - Select **Numeric** from the **Element Type** list.
10. Click **File**, **Save** and save the process as create_RegCodes_dimension.
11. Click **File**, **Execute** to create the RegCodes dimension.

Results

RegCodes is now available as a dimension in the Server Explorer.

Creating a RegName Dimension

Follow these steps to create the RegName dimension with a single string element.

Procedure

1. Open the Server Explorer.
2. In the Tree pane, right-click **Dimensions** and click **Create New Dimension**.

   The **Dimension Editor** opens.
3. Click **Edit**, **Insert Element**.

   The Dimension Element Insert dialog box opens.
4. Type **Name** in the **Element Name** field.
5. Select **String** from the **Element Type** list.
6. Click **Add**.

   The **Name element** now opens as a string element.
7. Click **OK**.
8. Click **Edit**, **Save** and save the dimension as RegName.

Creating the Translate Cube

Follow these steps to create the Translate cube.

Procedure

1. Right-click **Cubes** in the Server Explorer, and click **Create new cube**.

   The Creating Cube window opens.
2. Type **Translate** in the **Cube Name** field.
3. In the **Available Dimensions** box, double-click **RegCodes**.

   RegCodes moves to the Dimensions in new cube box.
4. In the **Available Dimensions** box, double-click **RegName**.

   RegName moves to the Dimensions in new cube box.
5. Click **OK** to save the two-dimensional Translate cube.

Populating the Translate Cube

Using the Cube Viewer, you can now enter the corresponding region names for the region codes.

The following steps illustrate how to populate the Translate cube with region names.

Procedure

1. In the **Tree** pane of the Server Explorer window, double-click **Translate**.

   The Cube Viewer opens.
2. Press F9 to see the elements in each dimension of the Translate cube.
3. Enter the region names that correspond to the region codes, using the table as your guide.

RegCode	**Name**
R54	Argentina
R32	Belgium

RegCode	Name
R55	Brazil
R1B	Canada
R56	Chile
R45	Denmark
R33	France
R49	Germany
R44	Great Britain
R30	Greece
R353	Ireland
R39	Italy
R352	Luxemburg
R52	Mexico
R31	Netherlands
R47	Norway
R351	Portugal
R34	Spain
R46	Sweden
R1A	United States
R598	Uruguay

4. Click **File**, **Close** to return to the Server Explorer.
5. In the Server Explorer, click **File**, **Save Data All** to save the cell values.

Creating the DBR Formula

You can now create a DBR formula that retrieves region names for each region code read into the processing worksheet.

Note: You must use DBR formulas, not DBRW formulas, in processing worksheets.

Procedure

1. In the processing worksheet, click cell B3.
2. Click **TM1** > **Edit Formula**.

   The Edit Formula bar opens.
3. Click **DB Ref**.

The Select Cube dialog box opens.

4. Click **Pick**.

   A different Select Cube dialog box opens.

5. Select **local:Translate** and click **OK**.

   TM1 correctly assumes that the element from the RegCodes dimension is in cell B1, but cannot find an element for RegName and shows it as undefined.

6. Click **RegName**.

   The Subset Editor opens.

7. Select the element **Name** and click **OK**.

   The regname field now displays Picked.

8. Click **OK** in the **Edit Reference to Translate** dialog box.

   The **Edit Formula** bar now displays the complete formula:

   ```
   =DBR("local:Translate", $B$1, "Name")
   ```

   This formula returns the value from the Translate cube found at the intersection of the Regcodes element in cell B1 and the Regname element Name.

9. Click **OK** to insert the formula in cell B3.

Results

Cell B3 now displays Argentina, which is the correct region for the code R54.

Creating Database Send (DBS) Formulas

You can create DBS formulas that send numeric data values to the cube because:

- You converted codes that map to elements in the Actvsbud dimension.
- You converted codes that map to elements in the Region dimension.
- You mapped multiple data values to their appropriate months.

Insert the DBS formulas in a row beneath the rows containing the data and mapping instructions. Do not insert them in the first row because they will be overwritten as TM1 reads in records into the processing worksheet.

Procedure

1. Click cell D4, an empty cell that will store the first DBS formula.
2. Click **TM1** > **Edit Formula.**

   The Edit Formula bar opens.

3. Click **DB Send**.

   TM1 prompts you to select the value to be sent to the cube.

4. Double-click cell D1, which contains the Jan cell value.

   TM1 prompts you to select the type of cell reference.

5. Click **Column Rel**.

   The DBS formula always references row 1, but the column reference will be relative to the location of the formula.

   TM1 prompts you to indicate the type of data in the cell.

6. Click **Numeric**.

   The Select Cube dialog box prompts you to select the cube to be populated.

7. Click **Pick**.

   A different Select Cube dialog box opens.

8. Select the **local:SalesCube** cube and click **OK**.

The **Edit Reference to Cube** dialog box displays with most of the mapping instructions for the value (cell D1) to be sent to the cube.

Cell A3 supplies an element in the Actvsbud dimension.

- Cell B1 supplies an element in the Region dimension.
- Cell C1 supplies an element in the Model dimension.
- Cell D3 supplies an element in the Month dimension.

To complete the mapping, you must identify an element for the Account1 dimension. Price.cma contains price values, so all DBS formulas should map to the price element.

9. Click **account1**.

   The Subset Editor opens.

10. Select **Price** and click **OK**.

    The account1 field of the Edit Reference to Cube dialog box now displays Picked.

11. Click **OK**.

    The Edit Formula bar displays the generated DBS function:

    ```
    DBS(D$1,"local:SalesCube",$A$3,$B$3,$C$1,"Price",D$3)
    ```

    For a full explanation of the formula, see “DBS Syntax” on page 92.

12. Click **OK** to place this formula in the processing worksheet.
13. Copy the formula in D3 to the range E3:O3.
14. Save the PriceProcessing worksheet.

DBS Syntax
The DBS function uses the following syntax:

```
DBS (value, server:cube, e1, e2[,...en]):
```

Argument	**Description**
value	Numeric value that is sent to the cube.
server:cube	The name of the cube that receives the sent value. The cube name must be prefixed with the name of the server on which the cube resides, for example sdata:SalesCube.
e1,...en	Elements that identify the cell location in the cube that receives the value. Specify the element arguments in dimension order. For example, e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension of the cube.

Processing a Data Source into a Cube

After you create a processing worksheet, you can process data into a cube.

If you have been completing the exercises in this section, you know that the PriceProcessing worksheet processes price values into the SalesCube cube.

You cannot use a processing worksheet to write values to cube cells that are calculated by rules, as you cannot edit rules-derived cell values.

- You must first verify that Price values in SalesCube are not derived by rules.
- Then you can process a data source into a cube.

Procedure

1. Open the Server Explorer.

2. Double-click the **SalesCube** cube.
3. Check if there is a rule attached to the cube. If not, skip to step 8.
4. If a rule is attached, open the rule in the **Rules Editor**.
5. Examine the rule to see if Price is calculated by rules.

   Note: Note that the rule includes the following statement which calculates the value for Price at both the numeric and consolidated levels:

   ```
   ['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);C:['Sales']\
   ['Units']*1000;
   ```

   This statement calculates the value for Price at both the numeric and consolidated levels.
6. Insert a pound sign (#) at the beginning of both statements to disable the calculation of Price.

   ```
   #['Price']=N:DB('PriceCube',!actvsbud,!region,!model,!month);#C:['Sales']\
   ['Units']*1000;
   ```

7. Save the rule.

   You can now use the processing worksheet to process the Price.cma source file into SalesCube.
8. If no rule is attached, open the processing worksheet that contains the DBS formulas and any mapping instructions.

   If you followed the earlier examples in this section, open the PriceProcessing processing worksheet.
9. Close any other worksheets.
10. Click **TM1** > **Process Data** > **Process**.

    The Select Cube, ODBC, or Flat File dialog box opens.
11. Click **Flat File**.

    The Select Input File dialog box opens.
12. Select the Price.cma sourcefile and click **OK**.

    TM1 processes the source file. During the process, a progress bar displays.

    TM1 sequentially reads each record of the source file into the first row of the processing worksheet. The processing worksheet recalculates after each record is read, and the DBS formulas send the values in the first row to the appropriate cell of the cube.
13. Browse SaleCube, and note that the Price values have been updated by the values in Price.cma.

Chapter 7. Controlling Access to TM1 Objects

This section describes how you can limit access to objects on an IBM Cognos TM1 server for all IBM Cognos TM1 installations, regardless of the authentication method.

Assigning Security Rights to Groups

You can assign object-level security for any non-administrative user group in TM1. By assigning security rights to groups, you can control a user's access to TM1 objects.

Note: You can not assign security rights for the ADMIN, DataAdmin or SecurityAdmin groups. The rights for these groups are predefined and appear disabled in the **TM1 Security Assignments** dialog box.

Note also that only the English versions of the security-level entries are accepted. The following keywords must be used as listed here when manually entering security levels in the TM1 cell security control cube.

The object-level security rights for TM1 groups are:

- `Admin` - Group has complete access to a cube, element, dimension, or other object.
- `Lock` - Group can view and edit a cube, element, dimension, or other object and can permanently lock objects to prevent other users from updating them.
- `Read` - Group can view a cube, element, dimension, process, or chore, but cannot perform operations on the object.
- `Reserve` - Group can view and edit a cube, element, dimension, or other object, and can temporarily reserve objects to prevent other users from updating them.
- `Write` - Group can view and update a cube, element, dimension, process, or chore.
- `None` - Group cannot see a cube, element, dimension, process, or chore, and cannot perform operations on the object.

The following table describes the security rights that you can assign to groups.

Privilege	Object	Description
`Admin`	Cube	Members of the group can read, write, reserve, lock, and delete the cube. They can save public cube views. They can also grant security rights to other users for this object.
	Element	Members of the group can access, update, reserve, lock, and delete the element. They can also grant security rights to other users for this object.
	Dimension	Members of the group can add, remove, and reorder elements in the dimension, and can reserve or lock the dimension. They can save public dimension subsets. They can also grant security rights to other users for this object.
	Application	Members of the group can see the application, use references within the application, and create both public and private references in the application. When a group has `Admin` privilege to an application, members of the group can set security privileges for all references and sub-applications within the application for other groups but not their own group.
	Reference	Members of the group can use the reference, as well as update or delete the reference. They can publish private references, and privatize public references.

© Copyright IBM Corp. 2007, 2017

Privilege	Object	Description
Lock	Cube	Members of the group have all privileges implied by Write permission, and can also lock the cube. When a cube is locked, nobody can update its data. The lock can be removed only by users who have Admin rights for the cube. Locks stays in place after the remote server shuts down.
	Element	Members of the group have all privileges implied by Write permission, and can also lock the element. When an element is locked, nobody can update cube cells identified by the element The lock can be removed only by users who have Admin rights for the element. Locks stays in place after the remote server shuts down.
	Dimension	Members of the group have all privileges implied by Write permission, and can also lock the dimension. When a dimension is locked, nobody can edit the dimension structure. The lock can be removed only by users who have Admin rights for the dimension. Locks stays in place after the remote server shuts down.
Read	Cube	Members of the group can see the cells in the cube, but cannot change their data.
	Element	Members of the group can see the cells identified by the element, but cannot change their data.
	Dimension	Members of the group can see the elements in a dimension, but cannot add, remove, or reorder the elements.
	Process	Members of the group can see the process in the Server Explorer, and can manually execute the process, but cannot edit the process. **Note:** Privileges assigned to processes are ignored when a process is executed from within a chore.
	Chore	Members of the group can see the chore in the Server Explorer, and can manually execute the chore, but cannot edit the chore.
	Application	Members of the group can see the application and use any public references within the application to which they have at least Read privilege. They can create private references in the application, and also create private sub-applications
	Reference	Members of the group can open the reference, but cannot update the reference in the application. Members of the group can, however, perform a "save-as" operation to save a new private version of the reference.

Privilege	Object	Description
`Reserve`	Cube	Members of the group have all privileges implied by `Write` permission, and can also reserve the cube to prevent other users from applying edits. The reservation can be removed either by the user who reserved the cube or by users who have `Admin` rights for the cube. A reservation expires automatically when the reserving user disconnects from the remote server or when the server shuts down.
	Element	Members of the group have all privileges implied by `Write` permission, and can also reserve the element to prevent other users from updating cube cells identified by the element. The reservation can be removed either by the user who reserved the element or by users who have `Admin` rights for the element. A reservation expires automatically, when the reserving user disconnects from the remote server or when the server shuts down.
	Dimension	Members of the group have all privileges implied by `Write` permission, and can also reserve the dimension to prevent other users from redefining the dimension. The reservation can be removed either by the user who reserved the dimension or by users who have `Admin` rights for the dimension. A reservation expires automatically when the reserving user disconnects from the remote server or when the server shuts down.
`Write`	Cube	Members of the group can read and update cells. They can save private cube views. The `Write` access privilege does not apply to cells identified by consolidated elements or to cells derived from rules.
	Element	Members of the group can read and update the cells identified by the element and edit attributes of the element.
	Dimension	Members of the group can edit element attributes, edit element formats, and create private subsets for the dimension. Members can also edit attributes for the dimension itself.

Privilege	Object	Description
None	Cube	Members of the group cannot see the cube in the Server Explorer, and thus cannot browse the cube.
	Element	Members of the group cannot see the element in the Subset Editor or Dimension Editor, and cannot see the cells identified by the element when browsing a cube.
	Dimension	Members of the group cannot see the dimension in the Server Explorer, and cannot browse a cube that contains the dimension.
	Process	Members of the group cannot see the process in the Server Explorer, and thus cannot execute the process. **Note:** Privileges assigned to processes are ignored when a process is executed from within a chore.
	Chore	Members of the group cannot see the chore in the Server Explorer, and thus cannot execute the chore.
	Application	Members of the group cannot see the application or its contents in the Server Explorer.
	Reference	Members of the group cannot see the reference in the Server Explorer.

Interaction of Different Object Security Rights

If you apply different security rights to the objects that identify a cell of data, TM1 applies the most restrictive security right to the cell.

Scenario 1

Suppose you assign a user Read access to the SalesCube cube, and `Write` access to the elements in this cube. In this scenario, the Read access of the cube overrides the `Write` access of the elements, and the user can view cube data but cannot update the cube data.

Scenario 2

The SalesPriorCube cube contains the following dimensions:

- Actvsbud
- Region
- Model
- Account1
- Month

Suppose a user has `Write` access to the SalesPriorCube cube, Read access to all of the elements in the Actvsbud dimension, and `Write` access to all of the elements in the other dimensions. The elements in the Actvsbud dimension identify every cell in the cube, and therefore the user cannot update any cube data.

Scenario 3

You can change the security rights for both cubes and dimensions. When groups have security rights for a cube, those rights apply to all dimensions in the cube, unless you further restrict access for specific dimensions or elements.

Suppose you want several regional groups of users to read all data in the SalesPriorCube cube. You also want each group to update data in its own region. For example, you want salespeople in the North America group to update North America data.

To implement this security scheme, you could:

- Create groups that reflect sales regions.
- Add users to the appropriate groups.
- Grant each regional group `Write` access to the SalesPriorCube cube.
- Grant the North America group Read access to those elements that do not reflect data for the North America region.

The TM1 sample data reflects this security scheme. Usr1 is in the North America group, which has `Write` access to the data associated with areas in the North America region, and Read access to the data associated with areas in other regions.

Securing Cubes

You can enhance or restrict a group's access to individual cubes. When you create a new cube, other groups initially have None access to the new cube. You must assign security rights to the new cube for other groups.

Assigning Security Rights for Cubes

Follow these steps to assign security rights for a cube.

Procedure

1. Open the Server Explorer.
2. Select the **Cubes** icon for the server you are working with.
3. Click **Cubes**, **Security Assignments**.

   The TM1 Security Assignments dialog box opens.
4. Click the cell at the intersection of the cube name and the group name for which you want to assign rights.

   You can assign rights for multiple cubes or to multiple groups by selecting a range of cells. To select a range of cells, click a cell to establish the top of the range, hold down Shift, and click further down the column or row to establish the bottom of the range.
5. Select the access level you want to assign.

   The name of the assigned access privilege displays in the cell.
6. Click **OK**.

Reserving and Releasing Cubes

When a user reserves a cube, that user gains exclusive rights to update the data in the cube. Other users cannot update the cube data until the cube is released. A cube can be released by either the user who reserved it or by a user who has `Admin` rights for that cube.

Consider reserving a cube as a way to temporarily freeze its data. A reservation expires automatically when the reserving user disconnects from the remote server or when the server shuts down.

Procedure

1. Open the Server Explorer.
2. Select the cube you want to reserve.
3. Click **Cube**, **Security**, **Reserve**.

   To release a cube:
4. Follow Steps 1 and 2 for reserving a cube.
5. Click **Cube**, **Security**, **Release**.

Locking and Unlocking a Cube

When a cube is locked, users cannot update the cube data or unlock the cube unless they are a member of the default `Admin` group. Members of user-defined groups who have `Admin` rights will not be able to unlock the cube. Unlocking a cube is restricted to members of the default `Admin` (or `DataAdmin`) group.

Consider locking a cube as a way to permanently archive its data. Locks stays in place after a server shuts down.

Procedure

1. Open the Server Explorer.
2. Select the cube you want to lock.
3. Click **Cube**, **Security**, **Lock**.

   To unlock a cube:
4. Follow Steps 1 and 2 for locking a cube.
5. Click **Cube**, **Security**, **Unlock**.

Securing Elements

You can enhance or restrict a group's access to individual elements using the **Element Security Assignments** dialog box.

Assigning Security Rights for Elements

Follow these steps to assign security rights for elements.

Procedure

1. Open the Server Explorer.
2. Select the dimension you want to work with.
3. Click **Dimension**, **Security**, **Elements Security Assignments**.

   The TM1 Security Assignments dialog box displays.
4. Click the cell at the intersection of the element name and the group name.

   You can assign rights for multiple elements or to multiple groups by selecting a range of cells. To select a range of cells, click a cell to establish the top of the range, hold down Shift, and click further down the column or row to establish the bottom of the range.
5. Select the access level you want to assign.

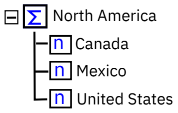
   The name of the assigned access privilege displays in the cell.
6. Click **Save** or **OK**.

   Note: If you click **Save**, you can continue to assign security rights to different elements. You can access elements in other dimensions by selecting a dimension in the **Select Dimension** field.

Interaction of Security Rights for Leaf and Consolidated Elements

You can set different levels of security for a consolidated element and the leaf elements that belong to the consolidation.

For example, the Region dimension in the sample data has the following element hierarchy:



Suppose Usr4 has Read access to the Canada leaf element and None access to the North America consolidated element. Usr4 can see the data identified by the Canada element, but cannot see the consolidated data identified by the North America element.

Reserving and Releasing Elements

When a user reserves an element, that user gains exclusive rights to update the data identified by that element. Other users cannot update the element's data, until the element is released. An element can be released by either the user who reserved it or by a user who has `Admin` rights for that element.

Consider reserving an element as a way to temporarily freeze the data that it identifies. A reservation expires automatically when the reserving user disconnects from the remote server or when the server shuts down.

Procedure

1. Open the Server Explorer.
2. Double-click the dimension you want to work with.

   The Subset Editor displays.
3. Select the element you want to reserve.
4. Click **Edit**, **Security**, **Reserve.**

   To release an element:
5. Follow Steps 1 through 3 for reserving an element.
6. Click **Edit**, **Security**, **Release**.

Locking and Unlocking an Element

When a user locks an element, only those users who have `Admin` rights for that element can update the data that it identifies. Even the user who locks the element cannot update its data, unless they have `Admin` rights for that element.

Consider locking an element as a way to permanently archive the data that it identifies. Locks stays in place after the remote server shuts down.

Procedure

1. Open the Server Explorer.
2. Double-click the dimension you want to work with.

   The Subset Editor displays.
3. Select the element you want to lock.
4. Click **Edit**, **Security**, **Lock**.

   To unlock an element:
5. Follow Steps 1 through 3 for locking an element.
6. Click **Edit**, **Security**, **Unlock**.

Securing Cells

Cell-level security applies to a specified cell and overrides all other TM1 security. Cell-level security requires:

- Creating a cell security control cube that contains a subset of the dimensions of a cube whose cell-level security you configure.
- Setting security for the appropriate cells in the security control cube by assigning security rights for TM1 security groups.

Note: For element-level security to apply to a cell, no security rights can be assigned to any TM1 security group for the cell. Cell-level security overrides element-level security, so cell-level security for the cell must be undefined.

Cell-level security applies to leaf elements and generally does not apply to consolidations, although you can use the None and Read security rights to control the display or editing of consolidations.

Creating a Cell Security Control Cube

Follow these steps to create a cell security control cube.

Procedure

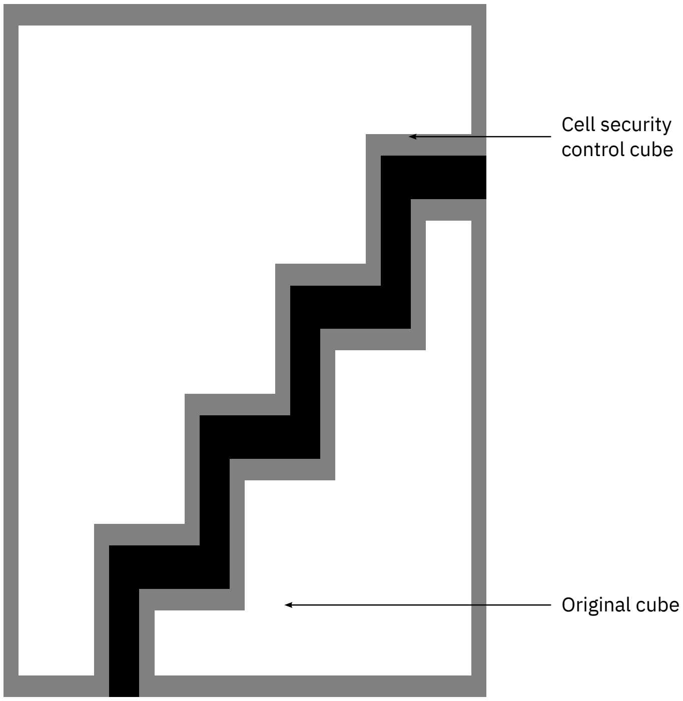
1. In TM1 Architect or TM1 Perspectives, right-click on the cube for which you want to define cell-level security and then select **Security**, **Create Cell Security Cube.**

   TM1 automatically creates a security control cube using the naming format }CellSecurity_*CubeName* where *CubeName* is the name of the cube that you selected. For example, if you selected the cube SalesCube, then TM1 creates the security control cube }CellSecurity_SalesCube.

   TM1 adds the dimensions of the original cube required to set security to the newly created security control cube, plus the }Groups dimension is added as the last dimension in the new cube.

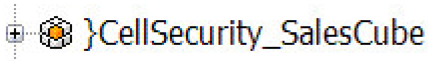
2. Click **View**, **Display Control Objects** if the control cubes are not already visible.

   TM1 displays the new security control cube along with the original cube.



   To apply security to cells in the security control cube, by TM1 security group:

3. Open the security control cube you just created by double-clicking the security control cube, for example,


}CellSecurity_SalesCube

4. Click **Recalculate** to display the security control groups, or click **Options**, **Automatic Recalculate**.
5. Expand the rows to display the cells to which you want to assign security rights.

   Note: Remember, cell-level security applies to leaf elements.

6. Enter the security level in the cube cells to assign security rights by user group.

   For details, see "Assigning Security Rights to Groups" on page 95.

   For example, the months in the second quarter for the Inspectors group have security assigned as None.

7. Close the security control cube.
8. Save the view.
9. Test the security levels by logging in as a user who is a member of the affected security group and viewing the cube for which you have set security.

Using Rules to Define Cell-Level Security

In the security control cube, you can use TM1 rules to apply cell-level security instead of entering security rights into security control cube cells.

For more information about using rules to apply cell-level security, see *TM1 Operations*.

Suppose you want to create a rule to apply cell-level security for the }CellSecurity_SalesCube cube. The following rule prevents any users in the Inspectors group from viewing any cells identified by the element Greece.

```
['Greece','Inspectors'] = S:'NONE';
```

Note: Make sure that security group names are unique and other dimensions or elements do not use the same name.

The benefits of using rules to implement security are:

- You do not enter security rights into cells in the security control cube, saving data entry time.
- Because TM1 does not store string rule values in memory nor saves them to disk, you save on memory consumption and disk storage.

Securing Dimensions

You can enhance or restrict a group's access to individual dimensions.

By default, TM1 security controls dimensions on the IBM Cognos TM1 server, as follows:

- Only members of the ADMIN and DataAdmin groups can create and delete dimensions on the TM1 server.
- Groups with Read access to a dimension can view dimension and element attributes through the Server Explorer, but cannot edit attribute values.
- Other groups initially have None access to new dimensions.
- When no security has been assigned to an element in a dimension, groups have `Write` access to new elements in that dimension.
- When you assign security rights to at least *one* element in a dimension, groups have None access to new elements in that dimension. Existing elements keep their original access (`Write`), unless you change that access.

Note: If you change the security in a dimension, and you want to reset that security to the default setting (groups have `Write`access to new elements added to the dimension), shut down your TM1 server and manually delete the }ElementSecurity<dimname>.cub file.

Assigning Security Rights for Dimensions

Follow these steps to assign security rights for a dimension.

Procedure

1. Open the Server Explorer.
2. Select the **Dimensions** icon for the server you are working with.
3. Click **Dimensions**, **Security Assignments**.

   The TM1 Security Assignments dialog box displays.
4. Click the cell at the intersection of the dimension name and the group name.

   You can assign rights for multiple dimensions or to multiple groups by selecting a range of cells. To select a range of cells, click a cell to establish the top of the range, hold down Shift, and click further down the column or row to establish the bottom of the range.
5. Select the access level you want to assign.

   The name of the assigned access privilege displays in the cell.
6. Click **OK**.

Reserving and Releasing Dimensions

When a user reserves a dimension, that user gains exclusive rights to add, remove, and reorder elements in that dimension. Other users cannot modify the dimension, until it is released. A dimension can be released by either the user who reserved it or by a user who has `Admin` rights for that dimension.

Consider reserving a dimension before you re-define it. A reservation expires automatically when the reserving user disconnects from the remote server or when the server shuts down.

Procedure

1. Open the Server Explorer.
2. Select the dimension you want to work with.
3. Click **Dimension**, **Security**, **Reserve**.

   To release a dimension:
4. Follow Steps 1 and 2 for reserving a dimension.
5. Click **Dimension**, **Security**, **Release**.

Locking and Unlocking a Dimension

When a user locks a dimension, only those users who have `Admin` rights for that dimension can add, remove, or reorder elements in that dimension. Even the user who locks the dimension cannot modify it, unless they have `Admin` rights for that dimension.

Consider locking a dimension if you want exclusive control of its definition.

Procedure

1. Open the Server Explorer.
2. Select the dimension you want to work with.
3. Click **Dimension**, **Security**, **Lock** from the pop-up menu.

   To release a dimension:
4. Follow Steps 1 and 2 for locking a dimension.
5. Right-click the mouse, and click **Dimension**, **Security**, **Unlock**.

Securing Processes

You can enhance or restrict a group's access to individual TurboIntegrator processes.

Note: TM1 ignores security rights assigned to TurboIntegrator processes when you execute a process from a chore. The security rights you assign to the chore determine the ability of a group to execute a process from a chore. For example, if a group has `None` access to Process1, but `Read` access to a chore that includes Process1, the group can execute Process1 from the chore.

Assigning Security Rights for Processes

Follow these steps to assign security rights for a process.

Procedure

1. Open the Server Explorer.
2. Select the Processes icon for the server you are working with.
3. Click **Processes**, **Security Assignments**.

   The TM1 Security Assignments dialog box opens.
4. Click the cell at the intersection of the process name and the group name.

   You can assign rights for multiple processes or to multiple groups by selecting multiple cells.

To select a range of adjacent cells, click a cell to establish the top of the range, hold down Shift, and click further down the column or row to establish the bottom of the range.

To select multiple non-adjacent cells, hold down CTRL, and click each cell.

5. Select the access level you want to assign.

   The name of the assigned access privilege displays in the cell(s).

6. Click **OK**.

Allowing Processes to Modify Security Data

The **TM1 Security Access** option controls whether a process is allowed to modify security data in the script of the process. Only members of the ADMIN and SecurityAdmin groups are allowed to set this option. This option is set on a process-by-process basis from the **Process** menu in Server Explorer.

When the **Security Access** option is enabled for a process:

- Members of the DataAdmin group are not allowed to edit the process since it may contain scripts that could modify TM1 security.
- Only members in the full ADMIN group can edit a process after the **Security Access** option has been enabled.
- Members of the SecurityAdmin group can view processes and turn this option on and off, but are never allowed to edit the contents of a process.
- The **View** option on the **Process** menu becomes available to allow users in the DataAdmin and SecurityAdmin groups to view processes in read-only mode.

For more details about the ADMIN, SecurityAdmin and DataAdmin groups, see *TM1 Operations*.

Enabling the Security Access Option for a Process

By default, the **Security Access** option is disabled for any new process. If you want to allow a new process to modify TM1 security, you need to manually enable the **Security Access** option for that process.

Procedure

1. In Server Explorer, select a process.
2. Click **Process**, **Security Access**.

   A check mark next to the Security Access option indicates that TM1 will allow the selected process to modify security data.

   Note: If you upgraded to TM1 9.4 or later from an earlier TM1 database, the Security Access option is automatically enabled for all of your existing TM1 processes. This allows your existing processes to continue running without requiring you to individually set the Security Access option for each process.

Viewing a Process in Read-only Mode

When the **Security Access** option is enabled for a process, users in the DataAdmin and SecurityAdmin groups can only view the process in read-only mode.

Procedure

1. In Server Explorer, select a process.
2. Right-click the process and click **View**.

   The process displays in read-only mode.

Securing Chores

You can enhance or restrict a group's access to individual chores.

Assigning Security Rights for Chores

Follow these steps to assign security rights for a chore.

Procedure

1. Open the Server Explorer.
2. Select the Chores icon for the server you are working with.
3. Click **Chores**, **Security Assignments**.

   The TM1 Security Assignments dialog box opens.
4. Click the cell at the intersection of the chore name and the group name.

   You can assign rights for multiple chores or to multiple groups by selecting multiple cells.

   To select a range of adjacent cells, click a cell to establish the top of the range, hold down Shift and click further down the column or row to establish the bottom of the range.

   To select multiple non-adjacent cells, hold down CTRL, and click each cell.
5. Select the access level you want to assign.

   The name of the assigned access privilege displays in the cell(s).
6. Click **OK**.

Securing Applications and References

You can assign security privileges for public items (references or sub-applications) within public TM1 applications to user groups on the IBM Cognos TM1 server. You must have `Admin` privilege to an application to assign security to items within the application.

You cannot assign security for private applications and references; only the user who creates a private item can access the item.

Procedure

1. In the Server Explorer, right-click the application that contains the items to which you want to assign security.
2. Click **Security**, **Security Assignments**.

   The TM1 Security Assignments window opens. The Name list contains all public items (TM1 objects, Excel files, and sub-applications) that reside in the current application.
3. Select the cell at the intersection of the item for which you want to define security and the user group to which you want to assign security.
4. Click one of the available security privileges.

   For details, see "Assigning Security Rights to Groups" on page 95.
5. Repeat steps 3 and 4 for any other items for which you want to define security.
6. Click **OK**.

   For any given user group, you can assign the `Read` or `Admin` privilege to a reference when you assign the `None` privilege to the source object associated with the reference. In this scenario, members of the user group cannot see the reference in an application.

   For example, if you assign the `None` privilege to a user group for a cube, but assign the `Read` privilege for a reference to the same cube, members of the user group cannot see the reference to the cube in an application.

Chapter 8. Using TM1 Action Buttons to Build Worksheet Applications

This section describes IBM Cognos TM1 Action button functionality which you can use to run processes and navigate between worksheets and Websheets.

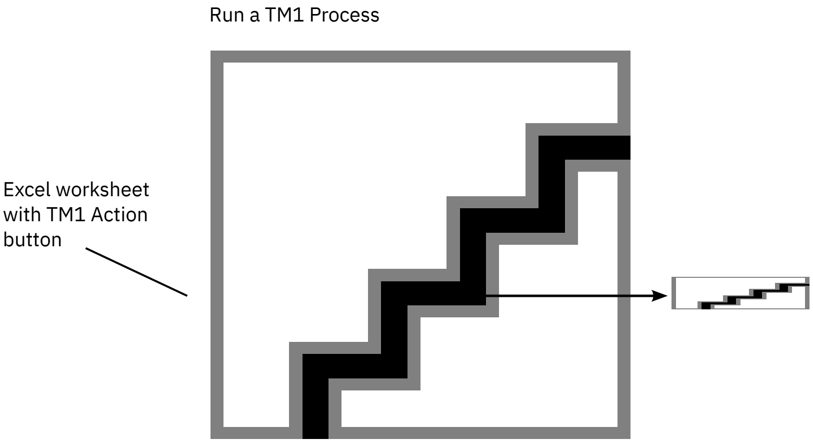
Overview

You can insert an Action button into a worksheet so users can run a TurboIntegrator process and/or navigate to another worksheet. Users can access these buttons when working with worksheets in Microsoft Excel with TM1, or with Websheets in TM1 Web.

An Action button can perform any of the following tasks:

- Run a TurboIntegrator process.
- Navigate to another worksheet.
- Run a TurboIntegrator process and then navigate to another worksheet.
- Recalculate a worksheet or rebuild the TM1 Active Form in a worksheet.

The following figure shows an example of an Action button in a worksheet.



Adding an Action Button to a Worksheet

You can insert an Action button into any empty cell in a worksheet using the following steps.

Procedure

1. In Excel, select an empty cell in your worksheet where you want to insert the Action button.

   Note: An Action button can not be inserted into a cell that contains data.

2. Select **Insert Action Button** from the **TM1** menu or click the Insert Action button on the TM1 toolbar.

   The Action button is inserted into the currently selected cell and the **Action Button Properties** dialog box appears.
3. In the **Action Button Properties** dialog box, click the **TM1 server** list to select the server where your data is located.

   If you want to dynamically retrieve the IBM Cognos TM1 server name from a cell or named-range whenever the Action button is run, click the **Use Reference** check box and then enter a cell or named-range reference.

   - To select a cell reference, click the Excel Reference button and then click the cell in the current worksheet where the server name is located.

© Copyright IBM Corp. 2007, 2017

- To retrieve the process name by referencing a named range in Excel, use the following format:

  ```
  =
  NameOfRange
  ```

  The named range must point to only a single cell that contains text for the server name.

  If you are not currently connected to the server that you want to use, click **Connect** to log in.

4. Click the **Action** option that you want the Action button to perform.

   You then need to configure the Action button depending on the type of action that you selected. For detailed steps, see the following sections:

Action	See
Run a TurboIntegrator Process	"Configuring an Action Button to Run a Process" on page 108
Go to another Worksheet	"Configuring an Action Button to Navigate to Another Worksheet" on page 111
Run a Process, then go to another Worksheet	"Configuring an Action Button to Run a Process and Navigate to a Worksheet" on page 113
Recalculate / Rebuild	"Configuring an Action Button to Recalculate or Rebuild a Worksheet" on page 113

5. Set the appearance properties of the Action button. See "Setting the Appearance Properties of an Action Button" on page 114.
6. To finish the Action button and return to your worksheet, click **OK** in the **Action Button Properties** dialog box.

Results

The Action button is updated in your worksheet and can now be used.

Configuring an Action Button to Run a Process

The following steps summarize how to configure an Action button to run a TurboIntegrator process.

Procedure

1. In the **Action Button Properties** dialog box, select the **Run a TurboIntegrator Process** option.

   The **Process** tab displays.
2. Select the process that you want to run. See "Selecting the Process to Run" on page 108.
3. Set the Process parameters. See "Setting Process Parameters" on page 109.
4. Select the calculation option that you want TM1 to apply before the Action button runs the process. See "Configuring an Action Button to Recalculate or Rebuild a Worksheet" on page 113.
5. Select the calculation option that you want TM1 to apply after the process has completed. See "Setting Process Options for Calculation" on page 110.
6. Configure the messages that you want TM1 to show before and after the process is run. See "Setting Process Options to Show Messages" on page 111.

Selecting the Process to Run

You can select the process you want to run in one of either two ways:

- Select the process name from the **Process** list.
- Dynamically retrieve the process name using an Excel reference.

Select the Process Name from the Process List

If you want to select the process name from a list of available processes on the current IBM Cognos TM1 server, perform the following steps.

Procedure

In the **Process** tab of the **Action Button Properties** dialog, click the **Process** list to select an available process from the TM1 server to which you are currently connected.

The Parameters grid opens and displays the parameters for the selected process.

Results

You then need to enter the parameter values for the selected process in the **Parameter** grid. See "Enter Parameter Values into the Parameter Grid" on page 109.

Use an Excel Reference to Retrieve the Process Name

If you want to dynamically retrieve the process name by referencing a cell or named range in the current worksheet, perform the following steps. The process name will be retrieved when the Action button is run.

Procedure

1. On the **Action Button Properties** dialog box, select **Get Process info from Worksheet** in the **Process** list.
2. Click the Excel Reference button next to the **Process Name** box to select a cell from the current worksheet.

   The **Select a Cell** dialog box opens.
3. Click the cell in the current worksheet where the process name is located.

   The location is automatically entered into the **Select a Cell** dialog box.
4. Click **OK** to close the **Select a Cell** dialog box.

   The cell reference appears in the **Process Name** box.
5. To retrieve the process name by referencing a named range in Excel, use the following format:

   ```
   =NameOfRange
   ```

   The named range must point to only a single cell that contains text for the process name.

Setting Process Parameters

Depending on how selected the name of the process to run, enter the parameter values in one of the following two ways.

- Enter parameter values into the **Parameter** grid.
- Create an Excel reference to retrieve the parameter values.

Enter Parameter Values into the Parameter Grid

If you selected the process name directly from the **Process** list, you can then enter the parameter values into the **Parameter** grid using the following steps. You can either type the values into the grid for each parameter, or use an Excel reference to dynamically retrieve a parameter value from the current worksheet when the Action button is clicked.

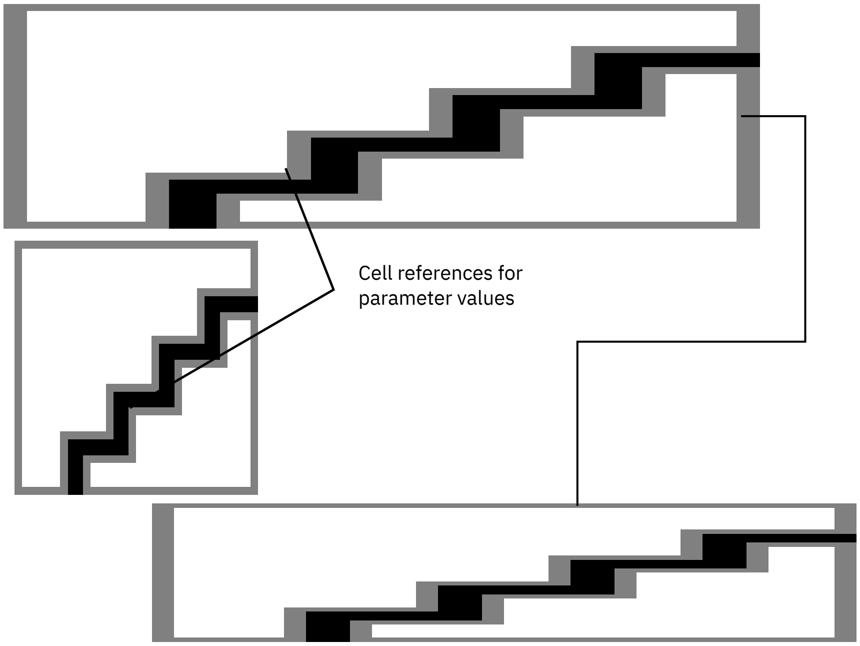
Procedure

1. In the **Process** tab of the **Action Button Properties** dialog, enter the parameter values into the **Parameters** grid.

   To directly enter the parameter values, type the values into the grid for each parameter.
2. To create a reference that dynamically retrieves a parameter value from the current worksheet, select the **Value** cell and then click .

   The **Select a Cell** dialog box opens.

3. Use the **Select a Cell** dialog box to select the cell in your worksheet where the parameter value can be found.



Use an Excel Reference to Retrieve the Parameter Values

If you selected **Get Process info from Worksheet** in the **Process** list, you must create an Excel reference that dynamically retrieves the process parameters from a worksheet.

Procedure

1. Click the Excel Reference button next to the **Parameters** box to select a reference from the current worksheet.

   The **Select a Range** dialog box opens.
2. Select the range of cells in your worksheet where the parameter values are located. Each cell must contain the value for only one parameter.

   Note: Parameters must be entered in the same order and type (string, numeric) as in the process.
3. Click **OK** to close the **Select a Range** dialog.

   The selected cell reference appears in the **Parameters** box.

   If you want to reference a named range in Excel, use the following format:

   ```
   =NameOfRange
   ```

   The named range must point to a single cell or a range of cells, depending on the parameters that the process is expecting.

   If the parameters for the process change, you must also update the process name and parameter settings for the Action button so the button can correctly run the process.

   Note: Due to a problem in Microsoft Excel if an Action button that runs a TurboIntegrator process with parameter inputs is created and saved in an Excel 2007 worksheet, the Action button stops working after the file is saved and then re-opened. As a workaround, when using an Action button in Excel 2007 to run a TI process that requires parameters, save the worksheet file in the .xls format.

Setting Process Options for Calculation

Use the **Process Options** dialog box to select the calculation operation that will be performed after the process is run.

Procedure

1. Click **Options** to show the **Process Options** dialog box.
2. Select the calculation operation that you want performed after the process has run. The available calculation options include:
   - **Automatically Recalculate Sheet** - Recalculates the values in the current worksheet.
   - **Rebuild Sheet** - Reloads the TM1 Active Form to its original report definition configuration.
   - **None** - The Action button will not perform any calculation or rebuild operation on the worksheet.

Setting Process Options to Show Messages

Use the **Process Options** dialog box to control the different message boxes that TM1 can show before and after the process is run.

Procedure

1. On the **Process** tab, click **Options** to open the **Process Options** dialog box.
2. Select the confirmation and status messages that you want TM1 to show.
   - **Show Success Message** - Displays a message after the process has successfully run.
   - **Show Failure Message** - Displays a message if the process does not run successfully.
   - **Show Confirmation Dialog** - Displays a confirmation message before the process is run. The user can click **Yes** or **No**.
3. Enter or edit the text for the messages you selected. You can also reference a cell or a named range to dynamically retrieve the message text. For example:
   - To retrieve the message text from the contents of cell A1 in the current worksheet, enter **=A1** into the message text box.
   - To reference a named range in Excel, use the format =**NameOfRange** .

     The named range must point to only a single cell that contains the text for the message.

Using the Action Button Server Name Property

The Action button Properties dialog box includes a field where you can enter a cell or named-range reference to dynamically retrieve the IBM Cognos TM1 server name for an Action button. To use this feature, click the **Use Reference** check box in the **TM1 server** section on the **Action Button Properties** dialog box and then enter a cell or named-range reference.

Configuring an Action Button to Navigate to Another Worksheet

You can use an Action button to navigate to another worksheet in the same workbook, or to a worksheet in another workbook.

The following steps summarize how to configure an Action button to navigate to another worksheet.

Procedure

1. In the **Action Button Properties** dialog box, click the **Go to another Worksheet** option.

   The **Worksheet** tab displays.
2. Select the target worksheet. See "Select a Target Worksheet" on page 112.
3. Set the **Match Title Elements** option. See "Enable the Match Title Elements Option" on page 112.
4. Set the **Replace Current Workbook** option. See "Setting the Replace Current Workbook Option" on page 112.
5. Select the calculation option that you want TM1 to apply before the Action button navigates to another worksheet. See "Configuring an Action Button to Recalculate or Rebuild a Worksheet" on page 113.
6. Select the calculation option that you want TM1 to apply after the navigation has completed. See "Setting Calculation Options for after Navigating to a Worksheet" on page 113.

Select a Target Worksheet

You can select a worksheet by choosing it from a list or by directly typing its name.

Procedure

1. Click the **TM1 Applications** option and then click **Browse**.

   The Select a Worksheet dialog opens.

   Note: To select a target workbook, you can also click the **Files** option and then click **Browse**. Then select a worksheet file from the **Open** dialog and then click **Open**. Another method to select a target workbook, is to click the **Excel Reference** button.
2. Select a worksheet and then click **OK** to return to the **Action Button Properties** dialog.

   The Application folder path and name of the worksheet you selected displays in the Workbook box.
3. To manually enter the Application folder path to the workbook, type the path into the **Workbook** box:
   - Start with the first folder name under Applications and use a back-slash \ character to separate folders. Do not include the Applications folder in the path. For example:

     ```
     Planning
     Sample\Bottom Up Input\Budget Input
     ```
   - To specify a worksheet and cell location to which you want to navigate, type the worksheet name and location in the **Sheet** box using the following format:

     ```

     =SheetName!ColumnNameRowName
     ```

     Note: If you enter a worksheet name that includes spaces, you must enclose the name in single quotes as shown in the following example.

     ='My First Sheet'!A2

Enable the Match Title Elements Option

The **Match Title Elements** option automatically matches and sets the text of the title dimensions in the target worksheet when a user clicks the Action button to navigate to the target worksheet.

When the **Match Title Elements** option is enabled, the dimensions in the source and target worksheets are automatically matched by the TM1 SUBNM and DBRW functions as follows:

- TM1 automatically matches title dimensions in the source and target worksheets based on the SUBNM formula in a cell.

  For example, when the same dimension exists in both the source and target worksheets, the element selected in the source worksheet is set for the same dimension in the target worksheet. When a column is selected in the source worksheet, it matches to the column with the same title dimensions in the target worksheet.
- TM1 automatically matches the row and column dimensions of the currently selected DBRW cell in the source worksheet to the matching title dimensions in the target worksheet, if they exist.

  For example, if Operating Expense and Feb-2004 are the row and column dimension elements for the currently selected element in the source worksheet, when navigating, these dimension elements are then matched to the title dimensions in the target worksheet. If the row Operating Expense and the column Feb-2004 are both selected in the source worksheet, the DBRW cell is the cell that exists in both the Operating Expense row and the Feb-2004 column. The row and column dimensions of the DBRW cell are then matched to title dimensions in the target worksheet.

Setting the Replace Current Workbook Option

The **Replace Current Workbook** option determines if the target worksheet is opened in a new window or in the same window, replacing the source worksheet.

For example:

- If **Replace Current Workbook** is not selected (default) and you are working in TM1 Web, then the source worksheet remains open and the target worksheet opens in a new tab.
- If **Replace Current Workbook** is selected and you are working in TM1 Web, then the source worksheet will be replaced by the target worksheet on the same tab and a new tab will not open.

Important: If you enable the **Replace Current Workbook** option, remember to save your workbook before testing the new button. You could lose your changes if you click the button and cause the current workbook to close.

Setting Calculation Options for after Navigating to a Worksheet

Set the calculation options to be applied to the target worksheet after navigating.

Procedure

On the **Worksheet** tab, select the **Calculation** options that you want to use.

- **Automatically Recalculate Sheet** - Recalculates the values in the current worksheet.
- **Rebuild Sheet** - Reloads the TM1 Active Form to its original report definition configuration.
- **None** - The Action button will not perform any calculation or rebuild operation on the worksheet.

Configuring an Action Button to Run a Process and Navigate to a Worksheet

To configure an Action that runs a TurboIntegrator process and then navigates to another worksheet, perform the following steps.

Procedure

1. In the **Action Button Properties** dialog box, click the **Run a Process, then go to a Worksheet** option.

   The **Process** tab displays.
2. Select the process that the Action Button will run. See "Configuring an Action Button to Run a Process" on page 108.
3. Select the worksheet to which the Action button will navigate. See "Configuring an Action Button to Navigate to Another Worksheet" on page 111.

Configuring an Action Button to Recalculate or Rebuild a Worksheet

You can use an Action button to perform only a recalculation or rebuild operation without running a TI process or navigating to a new worksheet. This can be useful if you only want to update the current sheet or reload the original version of an Active Form.

You can also use the **Calculate** tab to select the calculation operation that you want TM1 to perform before running a TI process or navigating to another worksheet.

Procedure

1. Insert an Action button into your worksheet as described in "Adding an Action Button to a Worksheet" on page 107.
2. On the **Action Button Properties** dialog box, click the **Calculate** tab.
3. In the **Calculate** tab, select the calculation option that you want to use.
   - **Automatically Recalculate Sheet** - Recalculates the values in the current worksheet.
   - **Rebuild Sheet** - Reloads the TM1 Active Form to its original report definition configuration.
   - **None** - The Action button will not perform any calculation or rebuild operation on the worksheet.
4. Set the appearance of the button as described in "Setting the Appearance Properties of an Action Button" on page 114.
5. On the **Action Button Properties** dialog box, click **OK** to close the dialog and insert the Action button into your worksheet.

Understanding Action Button Behavior with TM1 Active Forms

This section summarizes the behavior and order of operations when using an Action button with Active Forms.

Basic Action Button Behavior

Each time you click an Action button, the following steps are performed in this order:

1. The DBRW formula is captured for the currently selected cell in the source worksheet.

   The DBRW cell formula is captured before the following Recalc (F9) operation because the recalculation could change the number of active form rows due to zero suppression and/or MDX-based row subsets.
2. A Recalculation (F9) or rebuild operation is performed on the current worksheet, depending on the calculation option that was selected on the **Calculate** tab of the **Action Button Properties** dialog box.
3. The updated values are captured for any other cells that are referenced by advanced mapping options.

Additional Action Button Behavior

After the above basic steps have been completed, the following additional steps are performed, depending on whether the Action button is running a TI process, navigating to another worksheet or both.

When using an Action button to run a TI process only:

1. The TI process is run.
2. The calculation operation that was selected on the Action button's **Process Options** dialog box is performed.

When using an Action button to navigate only:

1. The navigation action begins.
2. Target values are set in the target worksheet.
3. The calculation operation that was selected in the **Worksheet** tab of the **Action Button Properties** dialog box is performed on the target worksheet.

When using an Action button to run a TI process and navigate to a worksheet:

1. The TI process is run.
2. The calculation operation that was selected on the Action button's **Process Options** dialog box is performed.
3. The navigation action begins.
4. Target values are set in the target worksheet.
5. The calculation operation that was selected in the **Worksheet** tab of the **Action Button Properties** dialog box is performed on the target worksheet.

Setting the Appearance Properties of an Action Button

Use the **Appearance** tab to set the caption, background picture, and other visual features for the Action button.

Click the **Appearance** tab to adjust the appearance properties of the button.

The Appearance tab has the following options:

- **Caption** - Sets the caption text that displays on the button.
- **Font** - Displays a standard font dialog where you can change the font type and size of the button text.
- **Show Background Image** - Allows you to select an image file (bmp, gif, or jpg format) that will be stretched to fit the button.
- **Display as Hyperlink** - When enabled, this option displays the button as a hyperlink with blue, underlined text instead of a standard button.
- **Preview** - This area shows an example of the button.
- **Colors** - Allows you to set text and background colors of the button. Click the color sample to display a Color dialog where you can select a standard color or define a custom color.

Using Advanced Navigation and Mapping Options

Use the **Advanced Options** dialog to manually map fields between the source worksheet and the target worksheet when inserting an Action button that navigates from one worksheet to another. This tool helps you map dimensions, cells, and values from the source worksheet to the target worksheet.

Note: Advanced mapping is applied after any automatic mapping has been performed by the **Match Title Elements** option.

To open the **Advanced Options** dialog, click **Advanced Options** on the **Worksheet** tab.

The **Advanced Options** dialog includes a grid where you define the mapping of fields between the source and target worksheets. Use the **Add** and **Delete** buttons to manage the rows in the grid.

Configuring Source to Target Mapping

You map the source worksheet to the target worksheet by setting values for the **Source Type**, **Source Object**, **Target Type**, and **Target Object** fields in the Advanced Mapping grid.

You can use the grid to specify how elements in the source and target worksheets get matched up when the target sheet opens. Each row in the grid defines one mapping configuration.

Use the following steps to configure advanced mapping:

Step	**See**
Indicate the type of object to map	"Set the Source Type" on page 115
Determine the value for the type of object you are using	"Set the Source Object" on page 115
Indicate the type of cell to map	"Set the Target Type" on page 116
Indicate where the value from the Source Object will be inserted	"Set the Target Object" on page 116

Repeat these steps to create more mapping configurations.

Set the Source Type

The **Source Type** field represents the type of object for the value you want to map.

Select the **Source Type** as follows:

- **SUBNM** - Indicates that you are mapping from a cell that contains a title dimension in the source worksheet.
- Selected DBRW - Indicates that you are mapping from a cell that contains a DBRW formula in the source worksheet.
- **Value** - Indicates that you will enter a string or numeric value that will be sent to the target.

Set the Source Object

The **Source Object** field takes a value, or Excel expression that evaluates to a value, depending on what is selected in the **Source Type** field.

Enter the **Source Object** as follows:

- If Source Type is set to **SUBNM**, then you need to specify the name of the title dimension that exists in the source worksheet.
- If Source Type is set to **Selected DBRW**, then you need to specify the name of a row or column title dimension that exists in the source worksheet.
- If Source Type is set to **Value**, then you need to enter a string or numeric value that will be sent to the target worksheet.

Note: You can also retrieve these values from the source worksheet by using the = symbol to create an Excel reference.

Set the Target Type

The **Target Type** is the type of cell in the target worksheet where the value from the **Source Object** field will be inserted.

Select the **Target Type** as follows:

- **SUBNM** - Indicates the target is a title dimension in the target worksheet.
- **Named Range** - Indicates the target is a named range in the target worksheet.
- **Range** - Indicates the target location is a cell in the target worksheet.

Note: If you set **Target Type** to either a **Named Range** or **Range**, any pre-existing data or formula in the target cell will be overwritten when you use the Action button to navigate. If the target cell contains a DBRW function, then the function will be lost and the cell will not be able to connect to, read from, or write to the IBM Cognos TM1 server.

Set the Target Object

The **Target Object** is the location in the target worksheet where the value from the Source Object will be inserted. This location can be either the name of a title dimension, a specific cell location, or a named range in the target worksheet, depending on what you selected for **Target Type**.

Enter the **Target Object** as follows:

- If **Target Type** is set to **SUBNM**, specify the name of the title dimension in the target worksheet. When **Target Type** is set to **SUBNM**, you must also enter a value for the **Subset** and **Alias** fields.
- If **Target Type** is set to **Named Range**, specify the name of the range in the target worksheet.
- If **Target Type** is set to **Range**, specify the cell location in the target worksheet.

You can enter a value for the **Target Object** by directly entering the location value or you can use an Excel reference to retrieve the location.

- **Directly enter value**

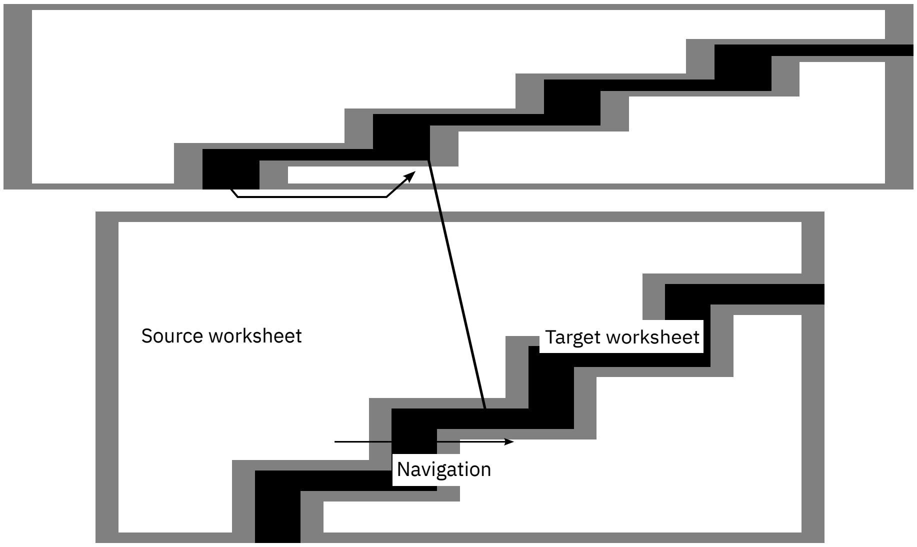
  To identify a location in the target worksheet, enter the value directly into the **Target Object** field without the = symbol. For example, enter C3 to identify the location of the **Target Object** as the cell C3 of the target worksheet.
- **Reference Excel cell**

  To reference a cell in the source worksheet that contains a location for the Target Object, include the = symbol. For example, the cell A1 in the source worksheet might contain the value C3 to represent the cell location for the Target Object in the target worksheet.

  Repeat all of the above steps to create more mapping configurations.

Example of Mapping a Source Value to a Target Cell

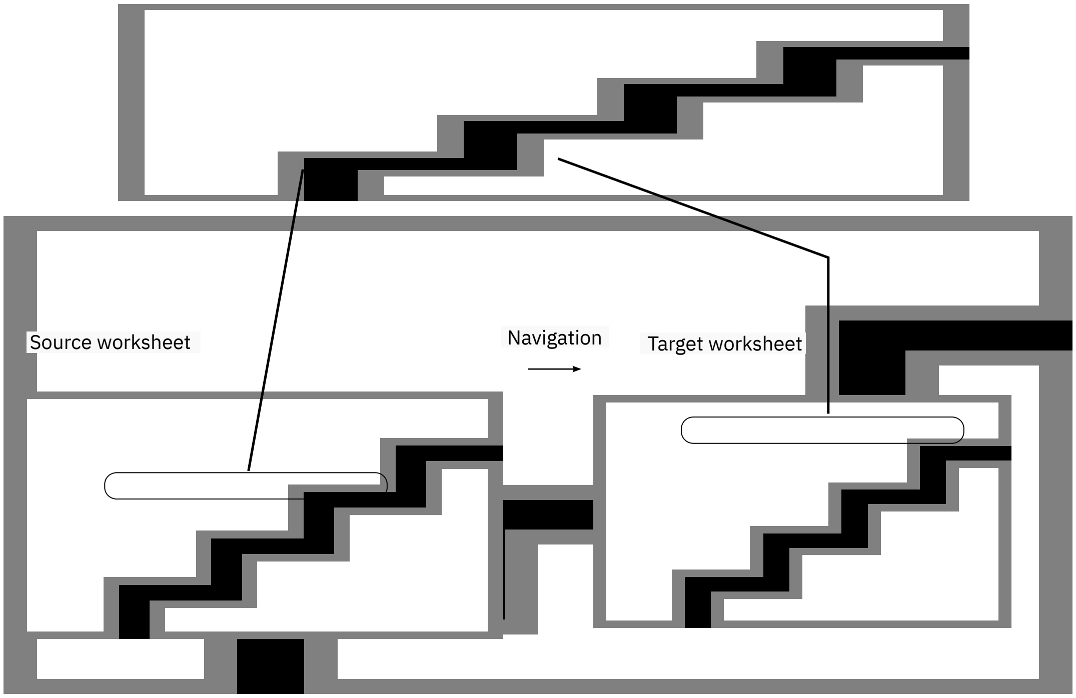
This example shows the mapping of a hard-coded value to one cell in the target worksheet.



Example of Mapping a Source SUBNM to a Target Cell

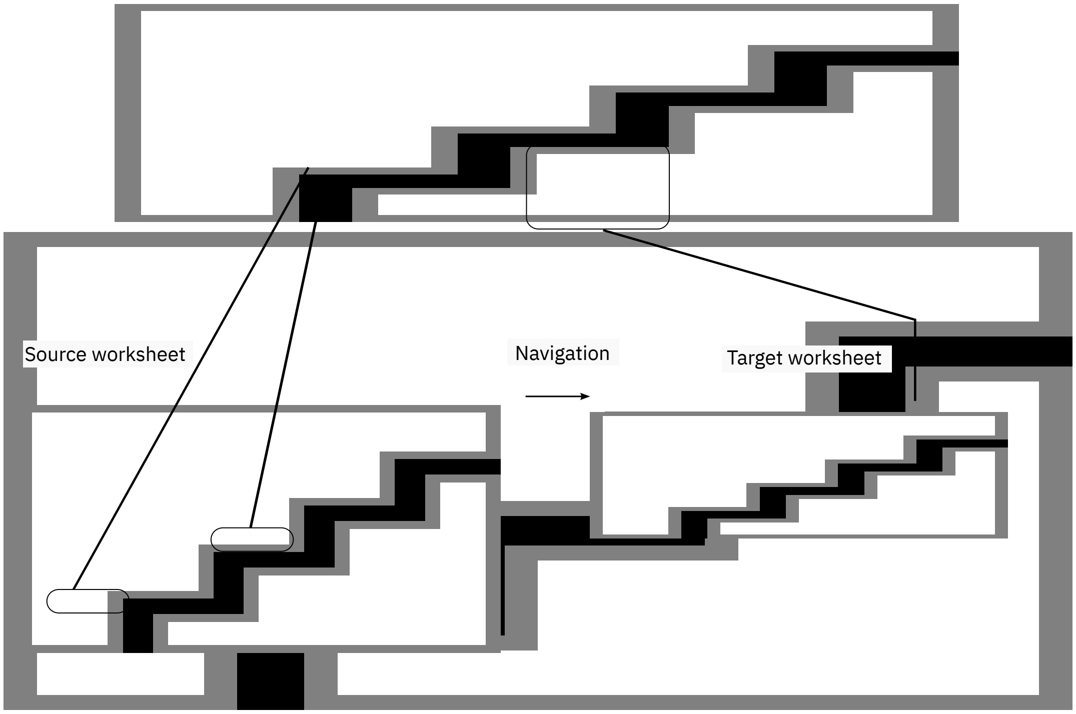
You can specify a SUBNM title dimension in the source worksheet to map to a corresponding SUBNM, named range, or range (cell) in the target worksheet.

For example, the following figure shows the S Series 2.0 L Wagon title element being inserted in the cell B2 in the target worksheet as a formatted caption.



Example of Mapping the Selected DBRW to the Target Worksheet

In this example, the row and column title dimensions for the selected DBRW cell are displayed in the target worksheet.



Modifying an Existing Action Button

After an Action button is inserted, you can edit its properties, size, and location as described in the following sections.

Editing the Caption, Background Image, and Properties of an Existing Action Button

To edit the properties of an Action button, right-click on the button and select an option from the menu.

The right-click menu provides the following main options:

- **Caption** - Opens the Button Caption dialog to edit the button's caption.
- **Background Image** - Opens a file selection dialog so you can browse and select an image to use as the background of the button.
- **Properties** - Opens the Action Button Properties dialog where you configure the actions to take place when a user clicks the button.

Moving and Resizing an Existing Action Button

After you click away from a new Action button, the button becomes enabled and is no longer in design mode. If you want to resize or move the button, you must turn on design mode in Excel as follows:

Procedure

1. In Excel, click **View**, **Toolbars**, **Control Toolbox**.

   The Excel Control Toolbox opens.

2. Click the **Design Mode** button.

   The text labels disappear on the Action buttons in the current worksheet.

3. Click on the **Action** button that you want to move or resize.

   Handles appear on the button indicating it can be resized and moved.

   - To resize the button, click and drag any of the handles.
   - To move the button, click and drag the button to a new location.

4. Exit design mode by clicking on the **Design Mode** button.

   The handles on the Action button disappear and the button returns to active mode.

Chapter 9. Using Data Reservations

This section describes all of the administrator, modeler, developer and programmer tasks related to enabling and implementing the Data Reservation feature available in IBM Cognos TM1.

Note: By default, Data Reservation is not enabled. An administrator must enable and configure the feature before you can use the related TurboIntegrator (TI) and API functions to manage Data Reservations.

Data Reservation Overview

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. Once reserved, the data in that region can only be modified by that specific user until the reservation is released.

You can use DR to support your specific business processes or to control data entry by acquiring and releasing Data Reservations on an as-needed or dynamic basis. For example, DR provides an alternative way to dynamically control write-access to cube data instead of trying to dynamically adjust TM1 security to achieve similar write-access restrictions.

Note: All DRs must be managed via custom applications that you develop using the provided TurboIntegrator (TI) and API functions. You design the application to obtain, release and manage DRs based on the required business process.

When to use Data Reservations

You would use Data Reservations if you want to manually or dynamically manage user write-access to TM1 data in a way that relates to your business process. Depending on your exact needs, you can apply the feature either as-needed or dynamically using TurboIntegrator and API functions.

For example, you might use DR to do the following:

- Manually run a one-time TurboIntegrator process that uses Data Reservations to lock out all users from making further edits to certain data after completing a specific project or closing a recent budget.
- Apply Data Reservations using TM1 Action buttons and TurboIntegrator processes in a Microsoft Excel spreadsheet or TM1 Websheet to dynamically control write-access to data as users interact with the data based on their roles and your company's business process.
- Use TM1 API functions within a custom external application to dynamically acquire and release Data Reservations.

How to use Data Reservations

To use Data Reservation, you must first enable and configure the feature for individual cubes and user groups and then use TurboIntegrator or API functions to programmatically apply and manage DRs.

To enable DR, use the following TM1 tools:

- **}CubeProperties control cube** - Enables and configures Data Reservation for individual *cubes*. For details, see "Enabling Data Reservation for cubes" on page 123.
- **Capabilities Assignments** - Determines if the members of a *user group* can manage (acquire and release) Data Reservations for themselves and other users. For details, see "Enabling user groups to manage Data Reservations" on page 124.

To apply and manage Data Reservations, you must use TurboIntegrator and TM1 API functions that allow you to programmatically obtain, release and manage reservations. For details, see the following topics:

- "Managing Data Reservations with TurboIntegrator functions" on page 126
- "Managing Data Reservations with TM1 API functions" on page 131

© Copyright IBM Corp. 2007, 2017

How Data Reservations interacts with other TM1 features

For details on how Data Reservations interact with other TM1 features such as data spreading, TurboIntegrator processes and security, see "Understanding Data Reservation behavior with other TM1 features" on page 121.

Security Overlay also works with DR. See Chapter 10, "Security Overlay," on page 137 for more information.

Data Reservation monitoring tools

You can use Server Explorer, the TM1Top utility, the TM1 Audit log, and certain TurboIntegrator and API functions to monitor Data Reservation activity and/or assignments. For details, see "Monitoring Data Reservations" on page 125.

Data Reservation modes

Data Reservation modes allow you to configure the Data Reservation feature depending on how you want to control write-access to your data. When you enable the Data Reservation feature for a cube, you choose one of the available Data Reservation modes.

The available Data Reservation modes include:

- Off (`OFF`)
- Required (`REQUIRED`)
- Required Shared (`REQUIREDSHARED`)
- Allowed (`ALLOWED`)

By default, Data Reservation is set to OFF for all cubes.

You configure the Data Reservation mode individually for each cube by entering the keyword for the mode in the cube's DataReservationMode property in the }CubeProperties control cube. For more details, see "DataReservationMode property" on page 124.

Required mode

The REQUIRED 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.

You set this mode by entering a value of `REQUIRED` in the }CubeProperties control cube for a specific cube.

Required Shared mode

The REQUIREDSHARED 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.

You set this mode by entering a value of `REQUIREDSHARED` in the }CubeProperties control cube for a specific cube.

REQUIREDSHARED mode was implemented to accommodate overlapping requests leveraging 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 server restricts write access to a slice by only allowing 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.

For example, REQUIREDSHARED mode can be used to allow access for multiple users to the same consolidated node. However, the application would then need to restrict access to the leaf nodes of the consolidation by assigning TM1 security rights to the related elements. An example of this is shown in the following hierarchy.

Table 2: Example hierarchy using REQUIREDSHARED mode for a Data Reservation

Consolidation:	**New England**	**User A**	**User B**
Leaf Nodes:	MA	WRITE access	READ access

Table 2: Example hierarchy using REQUIREDSHARED mode for a Data Reservation (continued)

Consolidation:	**New England**	**User A**	**User B**
	CT	WRITE access	READ access
	VT	WRITE access	READ access
	NH	READ access	WRITE access
	RI	READ access	WRITE access
	ME	READ access	WRITE access

An application can restrict write access to the leaf nodes MA, CT, and VT for User A, and to NH, RI, and ME for User B. This restriction can be enforced using element level security. Both User A and User B can then acquire a shared data reservation on the consolidation named New England. The reservation allows both users write access to the slice, while the underlying security restricts access to each user's set of leaf nodes.

Allowed mode

The ALLOWED mode 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 as needed.

For example, ALLOWED mode lets you use Data Reservation to set 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.

You set this mode by entering a value of `ALLOWED` in the }CubeProperties control cube for a specific cube.

Understanding Data Reservation behavior with other TM1 features

This section describes how Data Reservations (DR) interact with other TM1 features such as security, data spreading, TurboIntegrator (TI) processes and sandboxes.

Data Reservations and Security

Data Reservation is different from TM1 Security in the following ways

- DR does not override TM1 security assignments, but only adds another layer of write restriction on top of standard security for cube objects.
- DR applies to *individual* users while TM1 security applies to *groups* of users.
- DR applies restrictions only to cube data and individual users. It does not control any other TM1 objects.

This behavior is the same for both the REQUIRED and ALLOWED Data Reservation modes.

Data Reservations and the TM1 user interface

Data Reservation affects the appearance of cells in a cube view in all of the different TM1 user interfaces, such as Server Explorer, TM1 Perspectives and TM1 Web.

This behavior is different depending on the Data Reservation mode.

REQUIRED mode

When a cube is configured to use Data Reservation in REQUIRED mode, all of the cells appear with a gray background and are not writable except for the cells contained in your DR region. Only the cells in the DR for the current user will be writable and appear with a white background.

ALLOWED mode

When a cube is configured to use Data Reservation in ALLOWED mode, all of the cells are writable, depending on security, and appear with a white background except for the cells contained in the Data Reservations of other users. Cells reserved by other users appear with a gray background and are not writable.

For details on checking whether a cell is part of a DR, see "Displaying Data Reservation cell status in Server Explorer" on page 125.

Data Reservations and Data spreading

Data spreading behaves the same as before - cells that are not writable, such as calculated cells or cells with a hold, are skipped during a spread process. When a user has a Data Reservation and performs a data spreading operation, only the cells in that user's DR region will be considered writable and all other cells will be skipped.

This behavior is the same for both the REQUIRED and ALLOWED Data Reservation modes.

Data Reservations and TurboIntegrator processes and chores

You should understand the following considerations when using Data Reservation and also running interactive (non-scheduled) and scheduled TurboIntegrator (TI) chores/processes:

Some of this behavior is different depending on which Data Reservation mode is being used and whether the chore is run interactively or scheduled.

Interactive Processes and Chores

When a user interactively runs a process or a chore, for example from the TM1 user interface, then that process/chore runs as that user.

- For REQUIRED mode, this means that the process/chore can write only to data defined in the DRs held by that user.
- For ALLOWED mode, the process/chore can write to any cell that is either contained in a DR for that user or has the appropriate security rights for that user, but the process/chore cannot write to cells contained in another user's DR.

The following behavior is the same for both the REQUIRED and ALLOWED Data Reservation modes.

- If a write operation in the Interactive process/chore conflicts with the Data Reservation of another user, then the process/chore fails and an error message is displayed to the user.
- To run a process that acquires and releases DRs, the user running the process must belong to a user group that has the ManageDataReservation capability set to GRANT.

Scheduled Chores

When a scheduled chore automatically runs, it runs as the Admin user. Because of this, the chore may not be able to write to some cells if those cells are in the Data Reservation of another user.

When a scheduled chore encounters a Data Reservation conflict, the behavior is different depending on which Data Reservation mode is being used by the cube.

- In REQUIRED mode, if a write operation in a scheduled chore conflicts with a Data Reservation, the chore fails.

  To allow a scheduled chore to write to reserved cells when using DR in REQUIRED mode, you must handle this programmatically in your TurboIntegrator process by allowing the Admin user to write to the reserved cells. You can modify the TurboIntegrator process to acquire and release the necessary DRs for the Admin or possibly have the process temporarily turn off/on the DR feature for the applicable cubes.
- In ALLOWED mode, if a write operation in a scheduled chore conflicts with a Data Reservation, the chore will perform a data rollback and wait for the reservation to be released. When the reservation is released, the chore will retry the operation.

  You cannot cancel a waiting chore by changing the Data Reservation mode to REQUIRED or OFF. The chore will continue to wait until one of the following events occurs.

  - Release the Data Reservation that is blocking the chore so the chore can continue.
  - Use the TM1 Top utility to cancel the chore.
  - Shut down the server to cancel the chore.
  - Deactivate the scheduled chore.

Data Reservations and Sandboxes

You should understand the following considerations about obtaining and releasing Data Reservations on a specific cube while also entering data into sandboxes of that same cube.

DRs apply to the base version of a cube *and* any sandbox created from that cube.

The following sandbox behavior applies only to the REQUIRED Data Reservation mode.

- If the DR feature is enabled for a cube, then you must have a DR on that cube to write to the base data or any sandbox of that cube.

  For example, if a user has a DR to the Sales cube, then that user can only write to that same set of cells in any sandbox created from the Sales cube.
- If your DR for a cube has been released, then you can no longer write to any sandbox of that cube.
- You can commit a sandbox to base data after a DR is released, however, the edited values for any cells that are no longer reserved for you will be dropped with errors and your changes will be lost. To successfully commit the edited values to base data, you must first re-acquire the necessary DRs on that cube before attempting to commit the sandbox.

Data Reservations, Sandboxes, and Job Queue

If you are using Data Reservations with sandboxes and the Job Queue feature, the following behavior applies when committing sandbox data to base data, depending on which Data Reservation mode is being used by the cube.

- In REQUIRED mode, all necessary Data Reservations must be held until the queued operation is complete for the sandbox data to be successfully saved to base data. If the necessary Data Reservations are released before the Job Queue saves the data or if a Data Reservation conflict is encountered, the values in the conflicting cells will not be saved.
- In ALLOWED mode, if a sandbox commit in the Job Queue conflicts with a Data Reservation, the Job Queue will perform a data rollback for that operation and wait for the conflicting reservation to be released. All jobs in the Job Queue will be blocked until the waiting job is allowed to continue. The only way to unblock the Job Queue is to release the conflicting Data Reservations.

Enabling Data Reservation

By default, Data Reservation is not enabled. An administrator must enable and configure the feature before you can use the related TurboIntegrator (TI) and API functions to manage Data Reservations.

Before using Data Reservations, you must:

- Enable the feature for individual cubes (configured in the }CubeProperties control cube).

  For details, see“Enabling Data Reservation for cubes” on page 123
- Allow user groups the ability to manage, acquire and release DRs for themselves and other users (configured in the Capability Assignments window).

  For details, see “Enabling user groups to manage Data Reservations” on page 124).

After DR has been enabled for a cube, the feature applies to all users of that cube, but can only be managed by members of the user groups granted permission in the Capability Assignments window.

Enabling Data Reservation for cubes

Use the DataReservationMode property in the }CubeProperties control cube to enable or disable Data Reservation for a specific cube.

For more details about the }CubeProperties control cube, see the "Control Cubes" section in *TM1 Operations*.

Procedure

1. In Server Explorer, click the **View** menu and then click to select **Display Control Objects**.

   All of the TM1 Control cubes appear in Server Explorer with a prefix of a right curly brace }. For example, }CubeProperties.
2. In the Tree pane of Server Explorer, click to expand the **Cubes** node and then double-click the **}CubeProperties** control cube.

3. Enter a value for one of the available Data Reservation modes at the intersection of the DataReservationMode element (property) and the cube name to enable the Data Reservation feature for that cube.

DataReservationMode property

The DataReservationMode property uses the following values in the }CubeProperties control cube to configure the Data Reservation feature for individual cubes.

Enter these values into the }CubeProperties control cube using English only. These keyword values are not translated.

For more details about the differences between the different Data Reservation modes, see "Understanding Data Reservation behavior with other TM1 features" on page 121.

Value	Description
OFF	Turns off the Data Reservation feature for the specific cube. The default value is OFF.
REQUIRED	Sets the Data Reservation feature to REQUIRED mode for a specific cube. 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	Sets the Data Reservation feature to REQUIREDSHARED mode for a specific cube. 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 leveraging 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 server restricts write access to a slice by only allowing 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	Sets the Data Reservation feature to ALLOWED mode for a specific cube. This mode 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 as needed. For example, ALLOWED mode lets you use Data Reservation to set 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.

Enabling user groups to manage Data Reservations

Use the following capabilities in the TM1 Capability Assignments window to allow members of a user group to manage (acquire and release) Data Reservations for themselves and other users.

These capabilities are configured for an entire user group and not for individual users. A user must be a member of the group for the capability to apply.

Note: Only TM1 administrators have access to the Capability Assignments window.

For more details about Capability Assignments, see *TM1 Operations*.

Capability	Description
ManageDataReservation	Allows the members of the group to acquire and release Data Reservations. This capability is server-wide. Capabilities can not be applied to specific cubes or users. This capability is always set to GRANT for the standard ADMIN, DataAdmin, and SecurityAdmin groups and cannot be modified. Values for non-administrator groups: DENY - Default value for all non-administrator groups. GRANT- Enables this capability for a user group.
DataReservationOverride	Allows the members of the group to release reservations held by other users. This capability is always set to GRANT for the standard ADMIN, DataAdmin, and SecurityAdmin groups and cannot be modified. Values for non-administrator groups: DENY - Default value for all non-administrator groups. GRANT - Enables this capability for a user group.

Procedure

1. In Server Explorer, right-click on a server and select **Capability Assignments**.
2. In the Capability Assignments window, enter values for the **ManageDataReservation** and **DataReservationOverride** capabilities at the intersection with the user group you want to configure.

Monitoring Data Reservations

You can use the following tools to monitor Data Reservation activity and status.

- Server Explorer
- TM1 Top utility
- TM1 Audit Log
- TurboIntegrator and API functions

Displaying Data Reservation cell status in Server Explorer

You check the status of any cell in a cube view in Server Explorer as follows:

Procedure

In Server Explorer, open a cube view, right-click a cell and then click **Edit Status**.

A message box appears indicating whether the cell's value can be edited or that the cell is not editable because of a Data Reservation or other data access restriction.

Using TM1 Top to monitor threads waiting for Data Reservations

You can use the TM1 Top utility to monitor threads that are waiting for a data reservation to be released. Any thread in this state is shown in TM1 Top with a Data Reservation Release (DDR) value in the State field:

```
Wait:DRR
```

For more details about TM1 Top wait states, see the section "Understanding Thread Processing States" in the "System and Performance Monitoring" chapter in *TM1 Operations*.

Using the Audit Log to monitor Data Reservation events

If audit logging is enabled for the TM1 server, Cognos TM1 will record Data Reservation events. You can then use the TM1 Audit log to query and view the history of Data Reservation assignments.

For more details about using the TM1 Audit Log, see the "System and Performance Monitoring" chapter in *TM1 Operations*.

The following table lists the five Audit log events that are specific to Data Reservations.

Audit Log Event	Event Description
Property Set: CubeDataReservationEnable	Recorded when the value of the CubeDataReservationEnable property for a cube has changed.
Data Reservation: Acquired	Recorded when acquiring a Data Reservation.
Data Reservation: Released	Recorded when releasing a Data Reservation.
Data Reservation: Rollback Acquire	Recorded when rollback of a reservation acquire occurs.
Data Reservation: Rollback Release	Recorded when rollback of a reservation release occurs.

Notes® about Rollback Events

- The rollback of a Data Reservation acquire operation is to release the reservation.
- The rollback of a Data Reservation release operation is to re-acquire the reservation.

Querying for Data Reservation events

All Data Reservation events are associated with the TM1 cube object and are therefore grouped as cube events in the Audit log. You can query for these events in the Audit Log window as follows.

Procedure

1. In Server Explorer, open the Audit Log window.
2. In the **Event Type** section, select the **Object** option.
3. Set the **Object Type** field to **Cube**.
4. Set the **Event** drop-down list to one of the Data Reservation event types.
5. Run the query.

Using TurboIntegrator and API functions to monitor Data Reservations

As a custom solution, you could use the following TM1 TurboIntegrator and API functions to programmatically iterate cube objects and retrieve information about the related Data Reservations for each cube.

- TurboIntegrator - "CubeDataReservationGet" on page 129.
- TM1 API - "TM1DataReservationGetAll" on page 133.

Managing Data Reservations with TurboIntegrator functions

You can use the following TurboIntegrator (TI) functions to programmatically obtain, release and manage Data Reservations.

- CubeDataReservationAcquire
- CubeDataReservationRelease
- CubeDataReservationReleaseAll
- CubeDataReservationGet
- CubeDataReservationGetConflicts

Data Reservations are defined by a specific cube, user, and tuple (an ordered list of elements).

CubeDataReservationAcquire

CubeDataReservationAcquire acquires a Data Reservation for the specified cube, user and tuple.

This is a TurboIntegrator function, valid only in TurboIntegrator processes.

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.
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.

This is a TurboIntegrator function, valid only in TurboIntegrator processes.

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.

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.

This is a TurboIntegrator function, valid only in TurboIntegrator processes.

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.

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.

Argument	Description
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 is a TurboIntegrator function, valid only in TurboIntegrator processes.

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 is a TurboIntegrator function, valid only in TurboIntegrator processes.

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.

Argument	Description
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.

Managing Data Reservations with TM1 API functions

You can use the following TM1 C API functions to programmatically obtain, release and manage Data Reservations.

- TM1DataReservationAcquire
- TM1DataReservationRelease
- TM1DataReservationReleaseAll
- TM1DataReservationGetAll
- TM1DataReservationGetConflicts
- TM1DataReservationValidate

TM1DataReservationAcquire

Requests a DR for a specific IBM Cognos TM1 cube, user and tuple.

If there is an existing reservation owned by a different user whose region overlaps the requested reservation, then the reservation request will be rejected unless the bForce flag is used. If the bForce flag is true and the user running the API has the DataReservationOverride capability, then any conflicting reservations will be released and the new reservation will be granted.

Syntax

```
TM1DataReservationAcquire(TM1P hPool, TM1V hCube, TM1VhClient, TM1V bForce, TM1V
elementArray);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.

Parameter	Description
hClient	The owner to use for the reservation
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), then the function replaces any conflicting reservations.
elementArray	Array of element handles that define the tuple, the order must match the dimension order.

Return Value

Boolean value of true if the request was granted or false otherwise.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoReserveRights

TM1DataReservationRelease

Releases an existing DR for a specific IBM Cognos TM1 cube, user and tuple.

The owner used for hClient must match the holder of the DR for the command to succeed unless the user invoking the API has the DataReservationOverride capability enabled.

The addresses supplied must be an exact match.

Syntax

```
TM1DataReservationRelease(TM1P hPool, TM1V hCube, TM1V hClient,TM1V elementArray);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.
hClient	The owner of the reservation.
elementArray	Array of element handles that define the tuple. The order must match the dimension order.

Return Value

Boolean value of true if the request succeeded or false otherwise. Not finding the reservation is a failure and returns false. Insufficient privilege is handled as an error.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoAdminRights
- TM1ErrorObjectSecurityNoReserveRights

TM1DataReservationReleaseAll

Releases multiple Data Reservations for the specified IBM Cognos TM1 user.

The specified address tuple specifies the starting point for the search. All reservations owned by the specified user fully contained within the region defined by the address are released. Any reservation that overlaps the address but is not fully contained is not released.

Specifying a NULL client will remove reservations for all users. If the owner is not the same as the user executing the command, then the user must have the DataReservationOverride capability. Attempts to execute this command for a different user or all users without the override capability will be rejected without searching for existing reservations.

An administrator can release all reservations on a cube by specifying a NULL client and wildcards for every element in the address.

Syntax

```
TM1DataReservationReleaseAll(TM1P hPool, TM1V hCube, TM1V hClient,TM1V elementArray);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.
hClient	The owner of the reservation.
elementArray	Array of element handles that define the starting point for the release operation. The order must match the dimension order.

Return Value

Boolean value of true if there were no errors.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoReserveRights
- TM1ErrorObjectSecurityNoAdminRights

TM1DataReservationGetAll

Determines which Data Reservations are currently held on a IBM Cognos TM1 cube.

The client parameter is optional. If it is not supplied (the parameter is set to TM1ObjectNull), then all the DRs on the cube are returned.

If the client parameter is supplied, then only the DRs held by that particular user are returned.

Syntax

```
TM1DataReservationGetAll(TM1P hPool, TM1V hCube, TM1VhClient);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.

Parameter	Description
hClient	Optional handle for the user you want to query for.

Return Value

Array of DR data with the following format:

- [1] Cube name (TM1ValTypeString)
- [2-n] Array of DR information (TM1ValTypeArray)
  - [1] Creation Time
  - [2] User name (TM1ValTypeString)
  - [3-n] Array of element names defining the tuple (TM1ValTypeArray)
    - [1-n] Element name (TM1ValTypeString)

Possible Errors

TM1ErrorObjectHandleInvalid

TM1DataReservationGetConflicts

Determines which reservations currently held on a IBM Cognos TM1 cube will conflict with the specified client (user) and address.

This command can be used to gather the information needed to determine why an attempt to acquire a reservation failed, assuming the reservation that caused the denial is still there.

Syntax

```
TM1DataReservationGetConflicts(TM1P hPool, TM1V hCube, TM1V hClient, TM1V elementArray)
```

Parameter	Description
hPool	Standard memory pool used by all API commands
hCube	Handle to the cube you want to access
hClient	The client (user) to compare against current reservation owners.
elementArray	Array of element handles that define the tuple to compare against. The order must match the dimension order

Return Value

Returns an array of DR data with the following format:

- [1] Cube name (TM1ValTypeString)
- [2-n] Array of DR information (TM1ValTypeArray)
  - [1] Creation Time
  - [2] User name (TM1ValTypeString)
  - [3-n] Array of element names defining the tuple (TM1ValTypeArray)
    - [1-n] Element name (TM1ValTypeString)

TM1DataReservationValidate

Validates all the Data Reservations on a IBM Cognos TM1 cube.

Any reservation owned by a client (user) that no longer exists will be removed.

Syntax

```
TM1DataReservationValidate(TM1P hPool, TM1V hCube);
```

Parameter	Description
hPool	Standard memory pool used by all API commands
hCube	Handle to the cube we want to access

Return Value

Boolean value of true.

Possible Errors

TM1ErrorObjectNotFound (invalid cube)

API error codes for data reservations

The following table describes the possible error codes that can be returned by the IBM Cognos TM1 C API functions for data reservations.

Error	Description
TM1ErrorCubeNumberOfKeysInvalid	Number of elements doesn't match the number of cube dimensions.
TM1ErrorObjectHandleInvalid	Cube, Client, or Element handle does not map to an existing object.
TM1ErrorCubeKeyInvalid	The element supplied doesn't match an element in the dimension at that position. The element supplied is a UDC.
TM1ErrorObjectSecurityNoReserveRights	Capability to use reservation is not granted.
TM1ErrorObjectSecurityNoAdminRights	Attempt to release a reservation when not the owner and without the override capability being granted.

Chapter 10. Security Overlay

Security Overlay provides a mechanism to restrict users' ability to write to a cube, without causing contention on the dimensions of the cube and without needing to change underlying TM1 security. The effect of Security Overlay is to prevent updates to cell data by all users, except Administrators. As with Cell Security, Security Overlay you can define the restriction to only some of the dimensions of the data cube.

Security Overlay does not apply to an Admin user. The security overlay cube is considered to be a security cube so a TurboIntegrator process requires GrantSecurityAccess to modify it. Security overlay restrictions do not apply to the Admin user. This feature is different from privilege status (LOCK and RESERVE) which does apply to Admin.

The security overlay cube is created with a string prefix that identifies it as a security overlay cube in the same manner that cell security cubes are identified. `}SecurityOverlayGlobal_<Data Cube Name>`

The first N dimensions are the mapped dimensions from the data cube. The final dimension is the `}SecurityOverlay` dimension. This last dimension defines the data that is stored in the overlay cube. it has only one element. The `OverlayData` element stores the data that is used to implement the overlay. The OverlayData is where the values to restrict access go. The element is a string element. The `}SecurityOverlay` dimension is required because TM1 does not support cubes with only one dimension.

SecurityOverlayCreateGlobalDefault

This function is used to create or destroy a Security Overlay cube, and to set the overlay for a given area of a data cube.

Note that 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 is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

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');
```

© Copyright IBM Corp. 2007, 2017

SecurityOverlayDestroyGlobalDefault

This function is used to destroy a Security Overlay cube, and to set the overlay for a given area of a data cube.

Note that 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 is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

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

This function 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 is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

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 ‘|’.

Argument	Description
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.

Chapter 11. TM1 Web API

In addition to using IBM Cognos TM1 Web as a stand-alone application, you can also use it in your own custom web applications. Web programmers and TM1 application developers can use the Cognos TM1 Web application programming interface (API) to incorporate TM1 Web objects into custom web pages, applications, and dashboards.

The Cognos TM1 Web API includes two separate sets of APIs. These APIs also share a common login approach that uses session tokens or TM1 session IDs.

Depending on your specific development requirements, you can choose between the two different APIs and use the same login approach with either one.

Cognos TM1 Web API session login
The Cognos TM1 Web APIs share a common login approach that uses session tokens to uniquely identify and separate your Cognos TM1 Web sessions or TM1 session IDs to uniquely identify your TM1 Server. You can use this login approach with both APIs.
For more information, see "TM1 Web API session login" on page 141.

Cognos TM1 Web URL API
The URL API provides access to Websheet and CubeViewer objects by using a special set of URLs and parameters. Simple examples can be done right in the address bar of a web browser. To create a solution with the URL API, you need knowledge of HTML and an optional knowledge of JavaScript.
See "TM1 Web URL API" on page 147.

Cognos TM1 Web JavaScript Library
The JavaScript Library enables programmatic access to TM1 Web Websheet and CubeViewer objects in a combined HTML, JavaScript, and Dojo web page development environment. To use the JavaScript Library you need knowledge of HTML, JavaScript, Dojo, and the HTML Document Object Model (DOM).
See "TM1 Web JavaScript library" on page 168.

TM1 Web API session login

Use the session token login approach to uniquely identify your Cognos TM1 Web session. This login approach is recommended for the URL API. Use the TM1 session ID login approach to uniquely identify a Cognos TM1 server session, which might have multiple TM1 Web sessions. Use the session and login modules for easier session management with the JavaScript library.

Session token login

The session token login returns a unique session token that represents a login session for a specific user, Admin host, and TM1 server combination.

Important: Each TM1 Web session is associated with an HTTP session. The TM1 Web session token can be used only under the HTTP session in which it was created. You cannot save a TM1 Web session token, open a browser on another device, and get access to the TM1 Web session that corresponds to that session token because the HTTP session is different.

You can use the JavaScript `XMLHttpRequest` API to send an HTTP login request to the Cognos TM1 Web server. The session token is then returned in a JavaScript Object Notation (JSON) format from the request. After you receive the session token, you can then use it when you open TM1 Web objects.

If a timeout occurs with your HTTP session from inactivity, the Cognos TM1 Web session and related token are no longer valid.

TM1 Session ID login

Users can also log in by specifying a TM1 server session with a `TM1SessionId`. The TM1 server session that is used by a TM1 Web session will never change and must be generated or specified on creation. Multiple TM1 Web sessions can use the same TM1 server session.

© Copyright IBM Corp. 2007, 2017

Session and Login modules

In the JavaScript library, you can use the `session` and `LoginDialog` APIs to manage sessions and login dialog boxes.

For more information, see "Session and LoginDialog modules" on page 144.

Session token login

The overall process for logging in with a session token includes the following steps.

1. If you are using the URL API, set the `LegacyUrlApiSessionDiscoveryEnabled` configuration parameter in the `tm1web_config.xml` file.

   Note: This configuration parameter is not needed if you are using the JavaScript library.
2. Assemble a set of parameters for the login request that are based on the type of authentication you are using with Cognos TM1.
3. Post the login request to the Cognos TM1 Web server by using the JavaScript `XMLHttpRequest` API or other similar approach.
4. Process the JSON response to get the returned session token.
5. Use the session token when you open Websheet and CubeViewer objects.

Configuration parameter for session token login

If you are using the session token login approach with the URL API, you must set the `LegacyUrlApiSessionDiscoveryEnabled` configuration parameter in the `tm1web_config.xml` file to `False`.

This parameter enables the URL API session to be reused based on the specified admin host, TM1 server, and (optional) user name.

```
<add key="LegacyUrlApiSessionDiscoveryEnabled" value="False"/>
```

Login request parameters

Use the session token approach by sending a set of parameters in the request for the type of authentication that you are using with Cognos TM1.

For TM1 standard authentication and integrated login, use the following parameter format:

- param0=*TM1_Admin_host*
- param1=*TM1_server_name*
- param2=*username*
- param3=*password*

For example:

```
param0=localhost&param1=SData&param2=admin&param3=apple
```

If you are using IBM Cognos Business Intelligence security for authentication, use the following format to include a value for the *camPassport*:

- param0=*TM1_Admin_host*
- param1=*TM1_Server_name*
- param2=*camPassport*

JSON reply for session token login

The results of the login request are returned in a JSON formatted string.

If the login request is successful, the reply is returned in the following format.

```
{
    "reply":{
        "adminHost":adminHost,
        "sessionToken":sessionToken,
        "tm1Server":tm1Server,
        "username":username
    }
}
```

For example:

```
{
    "reply":{
        "adminHost":"localhost",
        "sessionToken":"06974cbd-ff2d-408b-8181-87bddd3f9048",
        "tm1Server":"Planning Sample",
        "username":"admin"
    }
}
```

If the login request is not successful, the following reply is returned.

```
{ "reply":null}
```

Example

The following example uses the JavaScript `XMLHttpRequest` API to post a login request to the TM1 Web server and retrieve the assigned session token.

```
<script type="text/javascript">

function login() {
   var xhr = new XMLHttpRequest();
   xhr.open("POST", "http://localhost:9510/tm1web/api/TM1Service/login", true);
   xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   xhr.onload = function() {
      var response = JSON.parse(xhr.responseText).reply;

      if(response != null) {
         var sessionToken = response.sessionToken;
         console.debug("Session token: " + sessionToken);
      }
      else {
         console.error("Login failed.");
      }
   }

   var params = "param0=localhost&param1=Planning+Sample&param2=admin&param3=apple";

   xhr.send(params);
};

</script>
```

LegacyUrlApiSessionDiscoveryEnabled configuration parameter

Use the `LegacyUrlApiSessionDiscoveryEnabled` configuration parameter to control how the TM1 Web URL API handles login sessions. Configure this parameter to specify whether or not the URL API tracks separate unique login sessions.

This parameter enables the URL API session to be reused based on the specified admin host, TM1 server, and (optional) user name.

If you are using the session token login approach with the URL API, you must set the `LegacyUrlApiSessionDiscoveryEnabled` configuration parameter in the `tm1web_config.xml` file to `False`. For more information about logging in with a session token, see "TM1 Web API session login" on page 141.

Format

```
<add key="LegacyUrlApiSessionDiscoveryEnabled" value=True or False/>
```

For example:

```
<add key="LegacyUrlApiSessionDiscoveryEnabled" value="False"/>
```

Values

The default value is `True`.

`True`
TM1 Web tries to match new login request with an existing login session based on the provided information (TM1 Admin host, TM1 Server, user name).

This parameter should only be set to `True` if a single login will occur for a unique TM1 Admin Host, TM1 server, and user name combination.

`False`
Specifies that a session token must be provided every time that you open a TM1 Web object with the TM1 Web URL API. Otherwise, the user is prompted.

Set this parameter to `False` if you plan to use multiple login sessions with TM1 Web URL API. You also use this configuration if you are using multiple login sessions with the URL API and other TM1 Web clients such as TM1 Web and TM1 Application Web. This configuration uses the session token to keep the user sessions separate and unique.

TM1 session ID login

Users can log in by specifying a TM1 server session with an admin host, TM1 server name, and `TM1SessionId`. The `TM1SessionId` corresponds to a user session on a TM1 server. To retrieve data from a TM1 server, a valid user session is required. Every TM1 Web session requires a TM1 server session. The overall process for logging in with a TM1 session ID is similar to the process for logging in with a session token except the `TM1SessionID` parameter replaces the `sessionToken` parameter:

`TM1SessionId=`*`valid TM1 session ID`*

This login method creates a new TM1 Web session and reuses the TM1 server session that corresponds to the `TM1SessionId`. If aTM1 server session is shared between TM1 Web sessions, invalidating the TM1 server session results in the TM1 Web sessions also being invalidated.

Example

In the following example, a `TM1SessionId` parameter is included in the URL to support this type of login authentication.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/Planning Sample/Bottom Up
Input/Budget Input&AdminHost=localhost&TM1Server=Planning Sample&TM1SessionId=<valid
TM1 session ID>
```

Session and LoginDialog modules

You can use the session and LoginDialog APIs to easily manage user sessions and login dialogs with the JavaScript library.

Session

You can use `tm1web/api/session/session` to retrieve information that is associated with the TM1 Web session. You can log in to, log out of, or retrieve information for a TM1 Web session.

Methods

login(params)
Performs a login to TM1 Web.
Parameters: `params` The login information object that uses one of the following object formats:

```
{
    adminHost: "localhost",
    tm1Server: "Planning Sample",
    username: "admin",
    password: "apple"
}
```

Or

```
{
    adminHost: "localhost",
    tm1Server: "Planning Sample",
    camPassport: "8sdf83uijsjdfsd903sd"
}
```

Or

```
{
    adminHost: "localhost",
    tm1Server: "Planning Sample",
    tm1SessionId: "D3lJLw50uvh2jtbAcIYyVA"
}
```

Returns `dojo/promise/Promise` as a promise that is resolved when the login action completes. If login fails, the promise is rejected, otherwise it is resolved. The promise is passed an object of the following format if login is successful.

```
{
    sessionToken: "7118fad5-bbeb-4b3e-8bea-4b4a45ca2735",
    tm1SessionId: "D3lJLw50uvh2jtbAcIYyVA",
    adminHost: "localhost",
    tm1Server: "Planning Sample",
     username: "Admin"
}
```

getInfo(sessionToken)

Retrieves the information that is associated with the TM1 Web session corresponding to the specified session token.
Parameters: `sessionToken` A session token corresponding to the TM1 Web session to retrieve information from.
Returns `dojo/promise/Promise` as a promise that is resolved when the action completes. If retrieval fails, the promise is rejected, otherwise it is resolved. The promise is passed an object of the following format if retrieval was successful.

```
{
    sessionToken: "7118fad5-bbeb-4b3e-8bea-4b4a45ca2735",
    tm1SessionId: "D3lJLw50uvh2jtbAcIYyVA",
    adminHost: "localhost",
    tm1Server: "Planning Sample",
    username: "Admin"
}
```

logout(sessionToken)

Performs a logout and invalidates the TM1 Web session corresponding to the specified session token.
Parameters: `sessionToken` A session token corresponding to the TM1 Web session to invalidate.
Returns `dojo/promise/Promise` as a promise that is resolved when the action completes. If retrieval fails, the promise is rejected, otherwise it is resolved. The action completes successfully even if the session does not exist or has already been invalidated.

For more information, see Dojo documentation for dijit._WidgetBase (https://dojotoolkit.org/reference-guide/1.10/dijit/_WidgetBase.html).

Examples

```
// login
require([
    "tm1web/api/session/session"
], function(session) {
    session.login({
        adminHost: "localhost",
        tm1Server: "Planning Sample",
        username: "admin",
        password: "apple"
    }).then(function(sessionInfo) {
        // Create Workbook or CubeViewer using sessionInfo.sessionToken
    }, function() {
        // Handle login failure appropriately
    });
});
```

```
// getInfo
require([
    "tm1web/api/session/session"
], function(session) {
    session.getInfo("sessionToken").then(function(sessionInfo) {
        // Continue using obtained sessionInfo
    });
});
```

```
// logout
require([
    "tm1web/api/session/session"
], function(session) {
    session.logout("sessionToken").then(function() {
        // Logout has successfully completed
    });
});
```

LoginDialog
You can use `tm1web/api/session/LoginDialog` to display or destroy a login dialog box.

Example

```
var dialog = new LoginDialog({
    onLogin: function(sessionInfo) {
        console.log(sessionInfo);
    },
    tm1Server: "Planning Sample",
    adminHost: "localhost"
});

dialog.show();
```

Construction

The `LoginDialog` module accepts several parameters for construction.

onLogin
Type: Function
Callback when login succeeds. An object that contains session information is passed as a parameter to the function on execution.

An example of this object is:

```
{
    tm1SessionId : "JcFxniSEzsJZVlQQhYDLDQ",
    sessionToken : "baa4ff9a-ddfb-41d1-9c71-f0add92325fd",
    adminHost : "localhost",
    tm1Server : "Planning Sample",
    username : "Admin"
}
```

This object has the same format as the response from the `login` method of `tm1web/api/session/session`.

adminHost
Type: String (optional)
Default value: localhost
The admin host from which to retrieve the TM1 server list. If no admin host parameter is specified, the AdminHost parameter value in the tm1web_config.xml file is used if it is specified.

tm1Server
Type: String (optional)
The TM1 server to log in to.

adminHostVisible
Type: Boolean (optional)
Default value: true
If false, the admin host text box is hidden from the login dialog.

tm1ServersVisible
Type: Boolean (optional)
Default value: true
If false, the list of TM1 servers are hidden from the login dialog

The `adminHost`, `tm1Server`, `adminHostVisible`, and `tm1ServersVisible` properties can be configured with the set method.

For example:

```
loginDialog.set("adminHost", "Planning Sample");
```

Methods

show()
Displays the login dialog box.

destroy()
Destroys the login dialog box.

For more information, see Dojo documentation for dijit._WidgetBase (https://dojotoolkit.org/reference-guide/1.10/dijit/_WidgetBase.html).

TM1 Web URL API

Use the Cognos TM1 Web URL API to include Cognos TM1 Web Websheet and CubeViewer objects in any HTML-based document or web page solution.

Cognos TM1 Web URL API overview

The URL API provides a framework for creating URLs that display Cognos TM1 Web Websheet and CubeViewer objects in your own custom web pages.

You can use the URL API to include Websheet and CubeViewer objects in any HTML-based solution such as web pages, web applications, and dashboards. The URL API provides access to Websheet and CubeViewer objects by using a special set of URLs and parameters.

Development tools

To create a solution with the URL API, you need knowledge of HTML and an optional knowledge of JavaScript.

For testing purposes and simple examples, you can use the URL API right in the address bar of a web browser. To create a solution with the URL API, you can use a simple text editor or any development environment that works with HTML and JavaScript.

The URL API uses HTML inline frames (`<iframe>` tag) as the primary way to display CubeViewer and Websheet objects in your custom web pages.

Features

You can assemble URLs that provide the following capabilities in your custom web pages:

- Websheet and CubeViewer
  - Access and display CubeViewer and Websheet objects
  - Set title dimension elements
  - Control properties such as turning the toolbar on or off
- CubeViewer
  - Display in either grid, chart, or grid and chart mode
  - Change chart type
  - Enable/disable auto-recalculation
  - Save the layout of a cube view
  - Recalculate the view
- Websheet
  - Rebuild Active Forms

Getting started with the Cognos TM1 Web URL API

You create a URL by using a base URL and specific TM1 parameters and then pass the completed URL to the TM1 Web server. The completed URL opens and displays a Websheet or CubeViewer object. You can also use the URL API to apply various actions on these objects.

The base URL and the parameters are separated by the hashtag symbol (#) and assembled in the following format:

*BaseUrl***#***Parameters*

If you want to include multiple parameters in the same URL, separate them with the ampersand symbol (&).

*BaseUrl***#***Parameter1=value***&***Parameter2=value***&***Parameter3=value*

Web browser address bar example

Copy and paste the following URL into the address bar of your web browser to see a simple example of the URL API.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
%20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning
%20Sample&Username=admin&Password=apple
```

Using the URL API in web pages

The URL API uses HTML inline frames (`<iframe>` tag) to display CubeViewer and Websheet objects in your custom web pages. The `<iframe>` tag is the primary way to display CubeViewer and Websheet objects with the URL API.

After a TM1 Web object is displayed in an iframe, you can then apply actions on that object by updating the `src` (source) property of the iframe with a new URL.

For more information, see "Using HTML <iframe> tags to display Cognos TM1 Web objects" on page 150.

Cognos TM1 Web URL API base URL

Use the base URL as the foundation for building all of your requests with the Cognos TM1 Web URL API.

An example of the base URL is shown in the following sample:

```
http://localhost:9510/tm1web/UrlApi.jsp
```

You combine the base URL with one or more parameters to make a complete request.

The base URL uses the following format:

```
http://WebServerName:PortNumber/tm1web/UrlApi.jsp
```

WebServerName

The domain name or IP address of the computer that is hosting the Cognos TM1 Web server.

For example, if you are working directly on the computer that is running Cognos TM1 Web server, you can use `localhost` for the *WebServerName* parameter.

```
http://localhost:9510/tm1web/UrlApi.jsp
```

If the TM1Web server is running on a remote computer, use the name of that system as follows:

```
http://MyWebServer:9510/tm1web/UrlApi.jsp
```

```
http://www.example.com:9510/tm1web
```

PortNumber

The port number for the web application server.

The standard TM1 installation uses a port number of 9510.

`UrlApi.jsp`

The capabilities of the Cognos TM1 Web URL API are provided through the `UrlApi.jsp` file.

Cognos TM1 Web URL API parameters

Parameters define which Cognos TM1 Web object you want to open and the actions to apply to those objects. You build a complete URL string by adding parameters to the base URL.

The base URL and the parameters are separated by the hashtag symbol (#) and assembled in the following format:

```
BaseUrl#Parameters
```

For example:

```
http://localhost:9510/tm1web/UrlApi.jsp#HideDimensionBar=true
```

If you include more than one parameter, separate them with the ampersand symbol (&).

```
BaseUrl#Parameter1=value&Parameter2=value&Parameter3=value
```

Note: Parameters are not case-sensitive. Either "`Action`" or "`action`" work the same, however capitalization is recommended for readability.

The most common parameters include `Action` and `Type`, which are used to open Workbook and CubeViewer objects. For example, the following URL shows an example of using parameters to open a CubeViewer object.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
%20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning%20Sample
```

After you open a Websheet or CubeViewer object in your web page, you can then use parameters to apply more actions to the object. For example, the following URLs use the `AutoRecalc` and `HideDimensionBar` parameters.

```
http://localhost:9510/tm1web/UrlApi.jsp#AutoRecalc=true
```

```
http://localhost:9510/tm1web/UrlApi.jsp#HideDimensionBar=true
```

For more information about working with parameters, see the following topics:

- "Using the Action parameter with TM1 Web objects" on page 153.
- "Using the Open parameter to open a TM1 Web object" on page 153.
- "Applying parameters and actions to an existing TM1 Web object" on page 154.

Using URL escape characters with the URL API

Use URL escape characters when creating URLs that contain spaces or other special characters.

Some common examples of URL escape characters include the following items:

Character	Escape Character
Space	%20
$	%24
%	%25
&	%26
=	%3D

Cognos TM1 Web URL API concepts

The basic concepts of using the URL API include displaying the objects in HTML iframes, specifying login credentials, opening objects, and applying actions.

Using HTML <iframe> tags to display Cognos TM1 Web objects

Use HTML inline frames (`<iframe>` tag) to display CubeViewer and Websheet objects with the URL API in your custom web pages.

The `<iframe>` tag is the primary way to display CubeViewer and Websheet objects in your custom web pages with the URL API.

After a TM1 Web object is displayed in an iframe, you can then apply actions on that object by updating the `src` (source) property of the iframe with a new URL.

Example

The following example uses a standard HTML button and a JavaScript function to load a Websheet into an iframe.

```
<!-- Button to load the websheet -->
<button onClick="loadWebsheet();">Load Websheet</button>

<!-- The iframe to host and display the Websheet -->
<iframe id="websheetId" style="width:100%; height:100%;"></iframe>

<script type="text/javascript">

   // The function to assemble the required URL and display the Websheet
   function loadWebsheet() {

      // Get a reference to the iframe
      webSheet = document.getElementById("websheetId");

      // Assemble the URL that specifies the Websheet you want to open
      baseUrl = "http://localhost:9510/tm1web/UrlApi.jsp";
      var websheetURL = baseUrl + "#Action=Open&Type=WebSheet";
      websheetURL = websheetURL + "&Workbook=Applications/Planning Sample/";
      websheetURL = websheetURL + "Management Reporting/Actual v Budget";
      websheetURL = websheetURL + "&AdminHost=localhost&TM1Server=Planning Sample";

      // Assign the URL to the iframe to display the Websheet
      webSheet.src = websheetURL;
   };
</script>
```

Specifying TM1 Admin Host and TM1 Server parameters with the URL API

You can set the Cognos TM1 Admin Host and server name in the URL string by using the `AdminHost` and `TM1Server` parameters.

The `AdminHost` and `TM1Server` parameters can be included in the URL with the `#Action=Open` command or implicitly specified with the use of a session token.

These values are optional in the URL, but must be provided to TM1 in one of the following ways.

- In the `tm1web_config.xml` file
- With a session token
- In the URL string
- Posted to the TM1 Web server using the form-based login
- Provided by the user when prompted by TM1 Web

If these values are not found, then TM1 will prompt the user for this information with a mini pop up window.

The Admin Host and server name are determined in the following order:

1. If a session token is specified, the admin host and TM1 server are determined from that first since it points to a specific session.
2. If the `AdminHost` and `TM1Server` parameters are set in the URL, they will override the values in the `tm1web_config.xml` file.
3. If these values are absent in the URL string, TM1 Web tries to determine if they are set in the `tm1web_config.xml` file.
4. If the `AdminHost` and `TM1Server` parameters are absent from both the URL string and the `tm1web_config.xml` file, then the system prompts the user for this information in a pop up window.

Example

These parameters use the following format:

`&AdminHost=`*`AdminHostName`*`&TM1Server=`*`TM1ServerName`*

where:

AdminHostName
Name of the system where the TM1 Admin Host is running.

TM1ServerName
Name of the TM1 server to log in to.

For example, the following sample code uses the local system and the TM1 Planning Sample database.

`&AdminHost=`**`localhost`**`&TM1Server=`**`Planning Sample`**

Managing user login and logout with the URL API

To view TM1 Web objects with the URL API, you must log in to the IBM Cognos TM1 server.

You can manage the user login process in the following different ways.

Session token login

The session token login tracks unique user sessions between multiple TM1 Web instances, TM1 Admin hosts, and TM1 servers.

The session token login is the recommended login approach. Use this login approach if your users log in to multiple instances of TM1 Web or separate TM1 servers at the same time.

For more information, see "TM1 Web API session login" on page 141.

TM1 Session ID login

Users can also log in by specifying a TM1 server session with an admin host, TM1 server name, and `TM1SessionId`. The `TM1SessionId` corresponds to a user session on a TM1 server. Every TM1 Web session requires a TM1 server session. The TM1 server session that is used by a TM1 Web session will never change and must be generated or specified on creation. Multiple TM1 Web sessions can use the same TM1 server session.

This login method creates a new TM1 Web session and reuses the TM1 server session corresponding to the `TM1SessionId`. If aTM1 server session is shared between TM1 Web sessions, invalidating the TM1 server session results in the TM1 Web sessions also being invalidated.

A TM1SessionId parameter can be included in the URL to support this type of login authentication. For example:

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/Planning Sample/Bottom
Up Input/Budget Input&AdminHost=localhost&TM1Server=Planning
Sample&TM1SessionId=<valid TM1 session ID>
```

Include user credentials in the URL

You can specify login information in the URL when you access TM1 Web objects. The URL must include values for AdminHost, TM1Server, UserName, or Password.

CAUTION: Specifying a password in the URL is not secure.

Pop-up login window

If all, or some, of the login information is not provided in any other way, then a pop-up window displays to prompt the user to log in before the Cognos TM1 Web objects can be displayed.

Form-based login

You can use a standard HTML form with input fields to collect a user's login credentials and post the information to the Cognos TM1 Web server. For more information, see "Cognos TM1 Web URL API form-based login" on page 152.

If you are using IBM Cognos Business Intelligence Security authentication, a CamPassport parameter can be specified.

Cognos TM1 Web URL API form-based login

You can use a standard HTML form with input fields to collect a user's login credentials and post the information to the Cognos TM1 Web server.

Make sure your form includes <input> fields with the following names. The field names and their related values are submitted to the Cognos TM1 Web server when you post the form.

- AdminHost
- TM1Server
- Username
- Password

Example

```
<!-- Login form -->
<form id="loginInfoForm" method="post">
   Admin Host: <input type="text" value="localhost" name="AdminHost" /><br>
   TM1 Server: <input type="text" value="Planning Sample" name="TM1Server" /><br>
   User Name: <input type="text" value="admin" name="Username" /><br>
   Password: <input type="password" value="apple" name="Password" /><br>
   <input type="button" value="Submit" onclick="loadCubeview();" />
</form>

<!-- The iframe to host and display the TM1 Web object -->
<iframe id="cubeviewId" name="cubeviewIFrame" style="width:100%; height:100%;"></
iframe>

<script type="text/javascript">

   // This function submits the login form and opens a CubeViewer
   function loadCubeview() {

      // Get a reference to the login form
      var loginForm = document.getElementById("loginInfoForm");

      var baseUrl = "http://localhost:9510/tm1web/UrlApi.jsp";

      var params = "#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan";
      params = params  + "&View=Budget Input Detailed&AccessType=Public";
      // Assign the URL to the action property of the login form
      loginForm.action = baseUrl + params;

      // NOTE: Be sure to use the iframe name for the target of the login form
      loginForm.target = "cubeviewIFrame";

      // Submit the form to login and display the TM1 Web object
      loginForm.submit();
   };
</script>
```

Logging out from the Cognos TM1 Web URL API

Use the `Action=Logout` parameter to end the current user session with the URL API.

You apply the logout action to an iframe that is already displaying a TM1 Web object. The logout action ends the session that opened that specific TM1 Web object and also ends the session for any other URL API instances under the same session.

The `Logout` action uses the following format:

`http://localhost:9510/tm1web/UrlApi.jsp#Action=Logout`

Example

The following example ends the session that is associated with the iframe and the related TM1 Web object.

```
function logout() {

   var baseUrl = "http://localhost:9510/tm1web/UrlApi.jsp";

   var webSheet = document.getElementById("websheetId");
   webSheet.src = baseUrl + "#Action=Logout";
};
```

Using the Action parameter with TM1 Web objects

The `Action` parameter specifies the type of action to perform on a TM1 Web object.

The most common action type is the `#Action=Open` command which can open either a CubeViewer or Websheet object.

Use the `Action` parameter in the URL string as follows:

`#Action=`*`TypeOfAction`*

The *TypeOfAction* value can be one of the supported actions such as `Open`, `Recalc`, or `Close`.

For a complete list of the available action types, see "URL API Action parameter" on page 162.

Example

For example, the following URL opens a TM1 Web CubeViewer object.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget Input
Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning Sample
```

Using the Open parameter to open a TM1 Web object

To open and display a TM1 Web object, use the `Action=Open` command and the `Type` parameter.

The `Open` parameter specifies that you want to open and display a TM1 Web object and the `Type` parameter specifies which type of object.

`Action=Open&Type=`*`object_type`*

The *object_type* can be either WebSheet or CubeViewer. Depending on the object type, additional parameters are required to specify the exact object to open. You can also set the title selection and other display properties in the same URL when you use the **Open** command.

For example, the following URL shows an example of using the Open and Type parameters to open a CubeViewer object.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
%20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning%20Sample
```

For more information about opening objects, see the following topics:

- "Displaying Websheet objects with the URL API" on page 155.
- "Displaying CubeViewer objects with the URL API" on page 156.

After you open a Websheet or CubeViewer object in your web page, you can then use parameters to apply more actions to the object. For more information, see "Applying parameters and actions to an existing TM1 Web object" on page 154.

Applying parameters and actions to an existing TM1 Web object

After a Cognos TM1 Web object is displayed in your web page, you can then use parameters to apply more actions to that specific object by updating the URL for the object.

To apply more actions to a Websheet or CubeViewer object that is already displayed, create a new URL with the parameters that you want. Then, apply the new URL to the src (source) property of the iframe where the object is displayed.

If the object is already displayed in an iframe, then you need to append only the action parameters to the base URL to create the new URL.

For example, the following URLs append the AutoRecalc and HideDimensionBar parameters to the base URL.

```
http://localhost:9510/tm1web/UrlApi.jsp#AutoRecalc=true
```

```
http://localhost:9510/tm1web/UrlApi.jsp#HideDimensionBar=true
```

Note:

The AutoRecalc parameter is only applicable to CubeViewer. It is not supported for Websheets.

In Websheets, auto recalculation is handled by the UseBookRecalcSetting parameter and the setting of the Excel workbook. For more information, see Cognos TM1 Web Configuration Parameters.

Example

The following example shows a JavaScript function that applies an updated URL to the src property of an iframe that is already displaying a CubeViewer object.

```
<!-- Use this iframe to display the CubeViewer (code not shown) -->
<iframe id="cubeviewId"></iframe>

<script type="text/javascript">
   // This function updates an existing CubeViewer object
   function toggleDimensionBar() {
      // Get a reference to the existing iframe and CubeViewer
      cubeView = document.getElementById("cubeviewId");

      // Create an updated URL and apply it to the iframe
      baseUrl = "http://localhost:9510/tm1web/UrlApi.jsp";
      cubeView.src = baseUrl + "#HideDimensionBar=True";
   };
</script>
```

Displaying Websheet objects with the URL API

A Websheet is a Microsoft Excel spreadsheet file that contains Cognos TM1 data and that you can view in a web browser. You can use the URL API to display a Websheet in an HTML iframe and then apply additional actions and parameters to the Websheet.

Opening a Websheet object

To open a Websheet object with the URL API, use the location path to the Websheet as organized in the TM1 Application folder.

Procedure

1. Open Cognos TM1 Web and expand the **Applications** node to locate the Websheet you want to open.
2. Build a text string that represents the path to the Websheet.

   Start the path with `Applications/` and separate any sub-folders with the forward slash / symbol.

   For example: `Applications/My Reports/Report_2014.xls`
3. Set the `Workbook` parameter in your URL to the path that you assembled.

   `#Action=Open&Type=WebSheet&Workbook=`**`Applications/My Reports/Report_2014`**
4. Combine the parameters with the base URL to make a complete URL request.

Example

Copy and paste the following URL directly into the address bar of your web browser to see this example.

```
http://localhost:9510/tm1web/UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/
Planning%20Sample/Management%20Reporting/Actual%20v
%20Budget&AdminHost=localhost&TM1Server=Planning%20Sample
```

The following JavaScript function loads a Websheet into an iframe.

```
function loadWebsheet() {

   // Get a reference to an existing iframe that has this ID
   webSheet = document.getElementById("websheetId");

   // Assemble the URL and assign it to the iframe
   webSheet.src = baseUrl + "#Action=Open&Type=WebSheet
   &Workbook=Applications/Planning Sample/Management Reporting/Actual v Budget
   &AdminHost=localhost&TM1Server=Planning Sample";

};
```

Setting display properties for the Websheet object

You can set the display property for the Websheet object by including the related parameter in your URL.

You can use the following parameter to change the display of a Websheet object:

HideToolbar
  Turns the toolbar on or off. The default is on.

Examples

Use the following format in your URL to control the display property of a Websheet object.

```
property=value
```

For example, add the following line to your URL to turn off the display of the toolbar.

```
HideToolbar=True
```

Selecting dimension title elements for Websheet objects

You can set the current elements in a title dimension of a Websheet object for any cell that contains a SUBNM function.

You can specify the dimension by either sheet number, row number, and column number or by dimension name.

You can select the new element by either element name or element index.

Format and values

Use the following format to specify the dimension by sheet, row, and column number:

Title_*S#-R#-C#=elementNameOrIndex*

Use the following format to specify the dimension by dimension name.

Title_*dimensionName=elementNameOrIndex*

Use the following parameters:

Title_S#-R#-C#

Specifies the title dimension by sheet number, row number, and column number.

Replace the # symbols with the values for the sheet, row, and column location of the dimension SUBNM cell in the Websheet.

Title_dimensionName

Specifies the title dimension by dimension name.

elementNameOrIndex

The string value for the name or numeric value for the index of the new title element that you want to select.

If you want to select the new title element by element index, instead of element name, include the UseIndex parameter in the URL as follows:

Title_S#-R#-C#=*ElementIndexNumber*&**UseIndex=true**

Example

Use the following example to first open a Websheet and then change the title element.

1. Copy and paste the following URL directly into the address bar of your web browser to first open the Websheet.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/Planning%20Sample/
Management%20Reporting/Actual%20v%20Budget&AdminHost=localhost&TM1Server=Planning
%20Sample
```

2. To change the title element, copy and paste the following URL into the same web browser session.

```
http://localhost:9510/tm1web/UrlApi.jsp#Title_S0-R11-C2=Canada
```

3. Copy and paste only the Title_S#-R#-C# parameter to the end of the base URL to get the similar results.

   Tip: Only the parameter section of the URL needs to be updated when you use parameters to apply changes. The base URL can remain unchanged.

```
Title_S0-R11-C2=US
```

4. Use the following sample with the UseIndex parameter to select a new title by element index.

```
Title_S0-R11-C2=3&UseIndex=true
```

Displaying CubeViewer objects with the URL API

The CubeViewer object displays the TM1 cube view in a custom web page. You can use the URL API to display a CubeViewer object in an HTML iframe and then apply additional actions and parameters to the object as needed.

Opening a CubeViewer object

To identify and open a Cognos TM1 Web CubeViewer object, combine the `Action=Open` command with the `Type`, `Cube`, `View` and `AccessType` parameters in your URL.

Use the following format to open a CubeViewer object:

```
#Action=Open&Type=CubeViewer&Cube=CubeName&View=ViewName&AccessType=Status
```

where:

- *CubeName* is the name of cube to which the view belongs.
- *ViewName* is the name of cube view.
- *Status* is the public or private status of the cube view. You must include a value of either `Public` or `Private` to correctly identify the specific cube view that you want to open.

Copy and paste the following URL directly into the address bar of your web browser to see this example.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
%20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning%20Sample
```

Use the following JavaScript function to load a CubeViewer into an iframe.

```
function loadCubeview() {

   // Get a reference to an existing iframe that has this ID
   cubeView = document.getElementById("cubeviewId");

   // Assemble the URL and assign it to the iframe
   cubeView.src = baseUrl + "#Action=Open&Type=CubeViewer
   &Cube=plan_BudgetPlan&View=Budget Input Detailed&AccessType=Public";

};
```

Setting display properties for the CubeViewer object

You can set the display properties for the CubeViewer object by including any of the related parameters in your URL.

You can use the following parameters to change the display of a CubeViewer object:

AutoRecalc

Turns automatic recalculation on or off. The default is off.

Note:

The AutoRecalc parameter is only applicable to CubeViewer. It is not supported for Websheets. Auto recalculation mode applies to gestures such as pivots, title changes, and zero suppression changes. Auto recalculation mode does not apply to data changes to leaf cells. Leaf cells always turn green when modified.

In CubeViewer, the AutoRecalc parameter serves the same purpose as the **Automatic Recalculation Mode** toolbar button (which does not exist for Websheets). When automatic recalculation mode is off (manual recalculation mode), gestures such as pivots, title changes, and zero suppression changes require a recalculation for data to be refreshed.

In Websheets, auto recalculation is handled by the `UseBookRecalcSetting` parameter and the setting of the Excel workbook. For more information, see Cognos TM1 Web Configuration Parameters.

HideDimensionBar

Turns the title bar on or off. The default is on.

Note: This setting applies to the CubeViewer object only.

HideToolbar

Turns the toolbar on or off. The default is on.

Examples

Use the following format in your URL to control the display properties of a CubeViewer object.

```
property=value
```

For example, add the following lines to your URL to change the display properties of the CubeViewer object.

```
AutoRecalc=False
```

```
HideDimensionBar=True
```

```
HideToolbar=True
```

Selecting title elements for the CubeViewer object

You can set the title elements in a CubeViewer object by adding the title parameter to your URL to specify the dimension and element name.

Use the following format and parameters:

`Title_DimensionName=ElementNameOrIndex`

Parameters:

DimensionName

The name of the title dimension that you want to change.

ElementNameOrIndex

The element name or the element index of the new title element you want to select.

If you want to select the new title element by element index, instead of element name, include the `UseIndex` parameter in the URL as follows:

`&Title_DimensionName=ElementIndex&UseIndex=True`

Example

Use the following example to first open a CubeViewer and then change the title element.

1. Copy and paste the following URL directly into the address bar of your web browser to first open the CubeViewer.

   ```
   http://localhost:9510/tm1web/
   UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
   %20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning%20Sample
   ```
2. To change the title element, copy and paste the following URL into the address bar of the same web browser session.

   ```
   http://localhost:9510/tm1web/UrlApi.jsp#Title_plan_version=FY 2003 Budget
   ```
3. Copy and paste only the parameter to the end of the base URL to update the title element.

   ```
   Title_plan_business_unit=Canada
   ```

   Tip: You only need to update the parameter section of the URL when using parameters to apply changes. The base URL can remain unchanged.
4. Try using the `UseIndex` parameter to select a new title by element index.

   ```
   Title_plan_business_unit=7&UseIndex=True
   ```

Displaying charts with the CubeViewer object

Similar to TM1 Web, the CubeViewer object can display TM1 data in grid-only, chart-only, or a combination of grid and chart mode. Use the `DisplayMode` and `ChartType` parameters to control the grid and chart display options.

Setting grid and chart display options

You can use the `DisplayMode` parameter to set the display of a CubeViewer object as grid only, chart only, or combined grid and chart.

The `DisplayMode` parameter uses the following format:

`DisplayMode=`*`value`*

The available options include the following values:

- `Grid`
- `Chart`
- `GridAndChart`

Example

`DisplayMode=`**`Chart`**

`DisplayMode=`**`Grid`**

`DisplayMode=`**`GridAndChart`**

Setting chart type with the URL API

Set the type of chart you want to display for a CubeViewer object by using the `ChartType` parameter.

The `ChartType` parameter uses the following format:

`ChartType=`*`ChartName`*

where *ChartName* can be the string value for one of the available chart types such as `Column` or `Pie`. For a complete list of available chart types, see "URL API ChartType parameter" on page 164.

URL example

Copy and paste the following URL directly into the address bar of your web browser to see this example.

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan&View=Budget%20Input
%20Detailed&AccessType=Public&AdminHost=localhost&TM1Server=Planning
%20Sample&DisplayMode=GridAndChart&ChartType=Pie
```

JavaScript example

```
<body>
<select title="Chart Type" onchange="setChartType(this.value);>
   <option></option>
   <option value="Point">Point</option>
   <option value="Bubble">Bubble</option>
   <option value="Line">Line</option>
   <option value="Spline">Spline</option>
   <option value="StepLine">Step Line</option>
   <option value="Bar">Bar</option>
   <option value="StackedBar">Stacked Bar</option    >
   <option value="Column">Column</option>
   <option value="StackedColumn">Stacked Column</option>
   <option value="Area">Area</option>
   <option value="SplineArea">Spline Area</option    >
   <option value="StackedArea">Stacked Area</option>
   <option value="Pie">Pie</option>
   <option value="Doughnut">Doughnut</option>
   <option value="Range">Range</option    >
   <option value="SplineRange">Spline Range</option>
</select>

<iframe id="cubeviewId" style="width:100%; height:100%;"></iframe>

<script type="text/javascript">
   function setChartType(value) {
      if(!value) {
         return;
      }

      cubeView = document.getElementById("cubeviewId");
      baseUrl = "http://localhost:9510/tm1web/UrlApi.jsp";
      cubeView.src = baseUrl + "#ChartType=" + value;
   };

</script>
</body>
```

Upgrading older URL API projects to the new Cognos TM1 Web 10.2.2 URL API

Use this information to upgrade your custom web pages that used the .NET-based Cognos TM1 Web URL API to the new Java-based Cognos TM1 Web 10.2.2 URL API.

As of IBM Cognos TM1 version 10.2.0, Cognos TM1 Web runs on a Java™-based web application server such as Apache Tomcat. Cognos TM1 Web version 10.2.0 does not require or use the Microsoft .NET Framework. Because of these changes, the URL API syntax and features are updated.

Changes in the Cognos TM1 Web 10.2.0 environment

Some of the main changes to Cognos TM1 Web are summarized in the following list. For more information about installation, configuration, and architecture, see *Planning Analytics Installation and Configuration*.

New default installation directory for TM1 Web

As of version 10.2.0, the default installation directory for Cognos TM1 Web is here:

`<TM1_install>\webapps\tm1web\`

New default URL for starting TM1 Web

Use the following new, default URL to open Cognos TM1 Web version 10.2.0:

`http://localhost:9510/tm1web/`

New TM1 Web configuration file and parameters

Cognos TM1 Web version 10.2.0 uses a new configuration file named `tm1web_config.xml`. This file replaces the `web.config` file from previous Cognos TM1 Web versions.

The new configuration file is location here:

`<TM1_install>\webapps\tm1web\web-inf\configuration`

Changes in the Cognos TM1 Web 10.2.2 URL API

The Cognos TM1 Web 10.2.2 URL API includes the following changes and updates:

Objects

- The Cognos TM1 Web Navigation tree object is not supported in the 10.2.2 URL API.
- The 10.2.2 URL API does not use the `ObjectId` parameter to track and apply actions on existing objects in your web page. Instead, the new URL API maintains the current state of the object internally for improved cross-domain usage. You can now apply additional actions on a TM1 Web object by using the iframe where the object is displayed.

Parameters

- Parameters are now separated from the base URL with the hash tag symbol (#) instead of the question mark (?).

  For example: `http://localhost:9510/tm1web/UrlApi.jsp#`*`Parameters`*
- The `OpenObject` parameter is renamed to `Open`.
- Parameter values of `Yes` and `No` are replaced with `True` and `False`. Values of 0 and 1 still work.
- The behavior of the `Action=Save` parameter in 10.2.2 is different and applies only to the CubeViewer object. This action saves only the layout of the view, and does not save changes to the data. Use the `Recalc` action to save data in a CubeViewer object.
- The `HideTitlebar` parameter is renamed to `HideDimensionBar`.
- The `HideTabs` parameter is no longer used.

- The ChartType parameter now uses string values instead of numeric values.

Required code changes for updating to the 10.2.2 URL API

To upgrade projects to the new URL API, review and apply the following code changes.

Change the base URL

Change your existing base URLs to use the new format for Cognos TM1 Web 10.2.2.

- Replace this URL: `http://`*`HostName`*`/TM1Web/TM1WebMain.aspx`
- With this URL: `http://`*`HostName`*`:9510/tm1web/UrlApi.jsp`

The UrlApi.jsp file replaces the TM1WebMain.aspx handler file.

Update the URL parameters

Review the list of changes in the Cognos TM1 Web 10.2.2 URL API.

For example, parameters are now separated from the base URL with the hash tag symbol (#) and some parameters are renamed.

Update the login process

The 10.2.2 URL uses a new session token login approach for uniquely identifying login sessions. A new form-based login is also available.

Replace the `ObjectId` parameter

Update your code anywhere you used the ObjectId parameter to track objects that you opened.

Instead, the new URL API maintains the current state of the object internally for improved cross-domain usage. Use this feature to apply more actions on a TM1 Web object by updating the src property of the iframe whenever you want to update an object.

Cognos TM1 Web URL API parameter reference

Use parameters to define which IBM Cognos TM1 Web object you want to open and the actions to perform on that object. You build a complete URL string by adding parameters to the base URL.

Note: Parameter format is show as &<parameter>=<value>. In examples, the parameter might appear as #<parameter>. The & is used to separate parameters, while # is used to mark the beginning of the parameters in examples.

URL API AccessType parameter

The AccessType parameter specifies the public or private status of the cube view that you want to display.

This parameter is used in combination the Action parameter when you open a CubeViewer object.

Format

&AccessType=*Value*

Values

Value	Description
Private	Specifies that the cube view has a private status.
Public	Specifies that the cube view has a public status.

Example

```
function loadCubeview() {
   cubeView = document.getElementById("cubeviewId");

   cubeView.src = baseUrl + "#Action=Open&Type=CubeViewer
    &Cube=plan_BudgetPlan&View=Budget Input Detailed
    &AccessType=Public
    &AdminHost=localhost&TM1Server=Planning Sample";
};
```

URL API Action parameter

The Action parameter specifies the type of action to perform on an IBM Cognos TM1 Web object.

Format

&Action=*Type_Of_Action*

Values

Value	Description
Close	Closes an existing object.
Logout	Ends the session for any other URL API instances under the same session.
Open	Opens a Cognos TM1 Web object.
Rebuild	Recalculates all values and rebuilds all subsets for a Cognos TM1 Active Form contained in a Websheet. This action performs the same action as when you click the Rebuild button on the Cognos TM1 Web toolbar.
Recalc	Recalculates an existing Websheet or CubeViewer object.
Reload	Reloads the CubeViewer object only.
Save	Saves the layout of a cube view. Applies only to CubeViewer objects. **Note:** The Save action does not save any changes to the data in the view. Use the Recalc action to save changed data.

URL Example

The following URL examples show some actions to perform on a CubeViewer or Websheet object that is already displayed in a web page.

http://localhost:9510/tm1web/UrlApi.jsp#Action=Save

http://localhost:9510/tm1web/UrlApi.jsp#Action=Reset

http://localhost:9510/tm1web/UrlApi.jsp#Action=Close

JavaScript Example

The following example shows a collection of JavaScript functions that each perform a different action on a CubeViewer or Websheet object.

```
<script type="text/javascript">

   function loadWebsheet() {
      webSheet = document.getElementById("websheetId");

      webSheet.src = baseUrl + "#Action=Open&Type=WebSheet
      &Workbook=Applications/Planning Sample/Management Reporting/Actual v Budget
      &AdminHost=localhost&TM1Server=Planning Sample";
   };

   function loadCubeview() {
      cubeView = document.getElementById("cubeviewId");

      cubeView.src = baseUrl + "#Action=Open&Type=CubeViewer&Cube=plan_BudgetPlan
      &View=Budget Input Detailed&AccessType=Public
      &AdminHost=localhost&TM1Server=Planning Sample";
   };
   function rebuildActiveForms() {
      webSheet.src = baseUrl + "#Action=Rebuild";
   };

   function recalculate() {
      getActiveIFrame().src = baseUrl + "#Action=Recalc";
   };

   function resetView() {
      cubeView.src = baseUrl + "#Action=Reset";
   };

   function saveView() {
      cubeView.src = baseUrl + "#Action=Save";
   };

   function close() {
      getActiveIFrame().src = baseUrl + "#Action=Close";
   };

</script>
```

URL API AdminHost parameter

The AdminHost parameter defines the name of the system where the IBM Cognos TM1 Admin Host is running. The default value is localhost.

Format

&AdminHost=*admin_host_name*

Values

The value of the AdminHost parameter is the name of the system where the Cognos TM1 Admin server is running.

Example

```
function loadCubeview() {
   cubeView = document.getElementById("cubeviewId");

   cubeView.src = baseUrl + "#Action=Open&Type=CubeViewer
   &Cube=plan_BudgetPlan&View=Budget Input Detailed&AccessType=Public
   &AdminHost=localhost&TM1Server=Planning Sample";
};
```

URL API AutoRecalc parameter

Use the AutoRecalc parameter to turn automatic recalculation on or off. The default is off.

Format

&AutoRecalc=*value*

Values

Value	Description
0, false	Disables automatic recalculation.
1, true	Enables automatic recalculation.

Example

```
function toggleAutoRecalcMode(enabled) {
        getActiveIFrame().src = baseUrl + "#AutoRecalc=" + enabled;
    };
```

URL API ChartType parameter

Use the `ChartType` parameter to set the type of chart that you want to display.

Format

&ChartType=*chart_type*

Values

Value	Chart Type
Point	Point
Bubble	Bubble
Line	Line
Spline	Spline
Stepline	Step Line
Bar	Bar
Stackedbar	Stacked Bar
Column	Column
Stackedcolumn	Stacked Column
Area	Area
Splinearea	Spline Area
Stackedarea	Stacked Area
Pie	Pie
Doughnut	Doughnut
Range	Range
Splinerange	Spline Range

Example

```
function setChartType(value) {
   if(!value) {
   return;
   }

   cubeView.src = baseUrl + "#ChartType=" + value;
   };
```

URL API Cube parameter

Use the `Cube` parameter to specify the name of cube to which the view belongs.

Format

&Cube=*cube_name*

Values

The value of the Cube parameter is the name of the cube, which contains the view that you want to open.

Example

```
http://localhost:9510/tm1web/UrlApi.jsp#Action=Open&Type=CubeViewer
&Cube=plan_BudgetPlan&View=Budget%20Input%20Detailed&AccessType=Public
&AdminHost=localhost&TM1Server=Planning%20Sample&DisplayMode=GridAndChart
&ChartType=Pie
```

URL API DisplayMode parameter

Use the `DisplayMode` parameter to display a CubeViewer object in grid, chart, or grid and chart mode.

Format

`&DisplayMode=`*`display_type`*

Values

Value	Description
`Chart`	Displays the CubeViewer object in chart-only mode.
`Grid`	Displays the CubeViewer object in grid-only mode.
`GridAndChart`	Displays the CubeViewer object with both a grid and chart.

Example

The following example shows a URL to apply to a CubeViewer object that is already displayed.

```
http://localhost:9510/tm1web/UrlApi.jsp#DisplayMode=Chart
```

The following example uses a JavaScript function to change the display mode.

```
function setDisplayMode(value) {
   if(!value) {
      return;
   }

   cubeView.src = baseUrl + "#DisplayMode=" + value;
};
```

URL API HideDimensionBar parameter

Use the `HideDimensionBar` parameter to control the display of the dimension title bar for the CubeViewer object. This setting applies to the CubeViewer object only.

Format

`&HideDimensionBar=`*`value`*

Values

Value	Description
`1`, `true`	Hides the dimension bar.
`0`, `false`	Displays the dimension bar.

Example

```
#HideDimensionBar=true
```

URL API HideToolbar parameter

Use the `HideToolbar` parameter to control the display of the toolbar for CubeViewer and Websheet objects.

Format

`&HideToolbar=`*`value`*

Values

Value	Description
`1`, `false`	Hides the toolbar.
`0`, `true`	Displays the toolbar.

Example

```
#HideToolbar=1
```

URL API TM1Server parameter

The `TM1Server` parameter specifies the IBM Cognos TM1 server to log into.

Format

`&TM1Server=`*`TM1_server_name`*

Values

The value of the `TM1Server` parameter is the name of the Cognos TM1 server to log in to.

Example

```
&TM1Server=Planning Sample
```

URL API TM1SessionId parameter

The `TM1SesionId` parameter specifies the IBM Cognos TM1 server to log into.

Format

`&TM1SessionId=`*`valid_TM1_session_ID`*

Values

Users can log in by specifying a TM1 server session with an admin host, TM1 server name, and `TM1SessionId`. The `TM1SessionId` parameter corresponds to a user session on a TM1 server.

For more information, see "TM1 Web API session login" on page 141.

Example

```
http://localhost:9510/tm1web/
UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/Planning Sample/Bottom Up
Input/Budget Input&AdminHost=localhost&TM1Server=Planning Sample&TM1SessionId=<valid
TM1 session ID>
```

URL API Type parameter

The `Type` parameter is used with the `Action` parameter to specify the type of object that you want to open.

Format

`&Type=`*`object_type`*

Values

Value	Description
`CubeViewer`	Defines the object as a CubeViewer.
`Websheet`	Defines the object as a Websheet.

Example

```
http://localhost:9510/tm1web/UrlApi.jsp#Action=Open&Type=CubeViewer
&Cube=plan_BudgetPlan&View=Budget%20Input%20Detailed&AccessType=Public
&AdminHost=localhost&TM1Server=Planning%20Sample
```

URL API View parameter

Use the `View` parameter to specify the name of the cube view that you want to open.

Format

`&View=`*`view_name`*

Values

The value of the `View` parameter is the name of the cube view.

Example

```
View=Budget%20Input%20Detailed
```

A complete URL is shown in the following example.

```
http://localhost:9510/tm1web/UrlApi.jsp#Action=Open&Type=CubeViewer
&Cube=plan_BudgetPlan&View=Budget%20Input%20Detailed&AccessType=Public
&AdminHost=localhost&TM1Server=Planning%20Sample
```

URL API Workbook parameter

The `Workbook` parameter specifies the path in the IBM Cognos TM1 server tree of the workbook to be loaded.

Format

`&Workbook=`*`path_to_workbook`*

Values

The value of the `Workbook` parameter is the path to the Cognos TM1 Websheet as organized in the TM1 Application folder.

Example

`&Workbook=Applications/Planning Sample/Management Reporting/Actual v Budget`

A complete URL is shown in the following example.

```
http://localhost:9510/tm1web/UrlApi.jsp#Action=Open&Type=WebSheet&Workbook=Applications/
Planning%20Sample/Management%20Reporting/Actual%20v
%20Budget&AdminHost=localhost&TM1Server=Planning%20Sample
```

TM1 Web JavaScript library

You can use the Cognos TM1 Web JavaScript library to programmatically access TM1 Web Websheet and CubeViewer objects in a combined HTML, JavaScript, and Dojo web page development environment. A working knowledge of JavaScript, Dojo Toolkit, and the HTML Document Object Model (DOM) is required for using the JavaScript library.

Overview

The Cognos TM1 Web JavaScript library includes the following main classes:

Workbook class
Represents a TM1 Web Websheet.

CubeViewer class
Represents a TM1 Web CubeViewer.

These main classes extend the Dojo Toolkit widget class called `dijit._WidgetBase`. This extension allows the Workbook and CubeViewer objects to be assigned as children of other Dojo objects such as a Dojo tab container or other container.

For more information about Dojo, see the Dojo documentation: http://dojotoolkit.org/documentation/.

The Websheet and CubeViewer objects also have a set of related properties and methods that you can access programmatically. These objects are loaded asynchronously and must finish loading before your code can interact with the objects.

Note:

In the Cognos TM1 Web JavaScript library, the following objects are deprecated:

- `tm1web/cubeview/CubeViewer`
- `tm1web/websheet/Workbook`

You should use `tm1web/api/CubeViewer` and `tm1web/api/Workbook` instead. The modules within the `tm1web/cubeview` and `tm1web/websheet` packages are now aliases of the modules within the `tm1web/api package`.

Configuration

The following configuration is required to use the Cognos TM1 Web JavaScript library.

1. Install Cognos TM1 Web and verify that you can log in to the standard user interface with a web browser.
2. Add the required references to the head section of your custom web page files that use the JavaScript library.

   For details, see “Required HTML <head> and <body> tags to use the JavaScript library” on page 169.

Getting started with JavaScript library

After you configured your Cognos TM1 Web environment, you can start coding your web pages to access objects with the JavaScript library. For more information and examples, see the following topics:

- “Loading Websheet objects with the JavaScript library” on page 172.
- “Loading CubeViewer objects with the JavaScript library” on page 174.

Configuring the AMD loader for the JavaScript Library

As of IBM Planning Analytics Local 2.0.0, it is no longer mandatory to use the version of Dojo that is provided with TM1 Web to load the TM1 Web JavaScript Library modules.

TM1 Web now supports using the AMD loader from Dojo version 1.7 and later to load the JavaScript Library modules.

For more information, see “Configuring the AMD loader for the JavaScript Library” on page 169.

Required HTML <head> and <body> tags to use the JavaScript library

The HTML <head> and <body> sections in each custom web page that uses the Cognos TM1 JavaScript library must include a set of required tags and references.

Add the following references to any of your HTML documents that use the JavaScript library.

- Include an HTML 5 DOCTYPE declaration.
- Add the `meta` references to the <head> section.
- Add the `class` reference to the <body> section.
- Add additional code to handle configuration of the AMD loader to find the JS Library modules correctly.

These references point to files contained under the Cognos TM1 Web installation directory.

TM1_Installation_Location`\webapps\tm1web\...`

Example

Use the following tags and references as a template.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
</head>
<body class="claro tm1web"></body>
</html>
```

Configuring the AMD loader for the JavaScript Library

You can use the AMD loader from Dojo version 1.7 and later to load the JavaScript Library modules.

Before any JavaScript Library module can be imported by using the AMD `require` function, the AMD loader must be configured to find and map the modules. The following example demonstrates how to configure the AMD loader for supported versions of Dojo.

Note: In the following examples, `location/to/tm1web/scripts/tm1web` represents the TM1 Web URI. An example of this location might be `http://localhost:9510/tm1web/scripts/tm1web`.

The following example shows how to configure the AMD loader for Dojo versions 1.8, 1.9 and 1.10.

```

require({
    packages: [
        {
            name: "tm1web",
            location: "location/to/tm1web/scripts/tm1web"
        },
        {
            name: "tm1webCom",
            location: "location/to/tm1web/scripts/com"
        },
        {
            name: "tm1webDojo",
            location: "location/to/tm1web/scripts/dojo"
        },
        {
            name: "tm1webDijit",
            location: "location/to/tm1web/scripts/dijit"
        },
        {
            name: "tm1webDojox",
            location: "location/to/tm1web/scripts/dojox"
        }
    ],
    map: {
        tm1web: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        },
        tm1webCom: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        },
        tm1webRave: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        },
        tm1webDojo: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        },
        tm1webDijit: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        },
        tm1webDojox: {
            dojo: "tm1webDojo",
            dijit: "tm1webDijit",
            dojox: "tm1webDojox",
            com: "tm1webCom"
        }
    }
});
```

The following example shows how to configure the AMD loader for Dojo 1.7.

```
require({
    packages: [
        {
            name: "tm1web",
            location: "location/to/tm1web/scripts/tm1web",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            }
        },
        {
            name: "tm1webCom",
            location: "location/to/tm1web/scripts/com",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            }
        },
        {
            name: "tm1webRave",
            location: "location/to/tm1web/scripts/com",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            }
        },
        {
            name: "tm1webDojo",
            location: "location/to/tm1web/scripts/dojo",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox"
            }
        },
        {
            name: "tm1webDijit",
            location: "location/to/tm1web/scripts/dijit",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox"
            }
        },
        {
            name: "tm1webDojox",
            location: "location/to/tm1web/scripts/dojox",
            packageMap: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox"
            }
        }
    ]
});
```

The following example shows a complete configuration.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<script src="path/to/the/1.10/version/of/dojo.js"></script>
<script>
    require({
        packages: [
            {
                name: "tm1web",
                location: "http://localhost:9510/tm1web/scripts/tm1web"
            },
            {
                name: "tm1webCom",
                location: "http://localhost:9510/tm1web/scripts/com"
            },
            {
                name: "tm1webDojo",
                location: "http://localhost:9510/tm1web/scripts/dojo"
            },
            {
                name: "tm1webDijit",
                location: "http://localhost:9510/tm1web/scripts/dijit"
            },
            {
                name: "tm1webDojox",
                location: "http://localhost:9510/tm1web/scripts/dojox"
            }
        ],
        map: {
            tm1web: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            },
            tm1webCom: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            },
            tm1webRave: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            },
            tm1webDojo: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            },
            tm1webDijit: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            },
            tm1webDojox: {
                dojo: "tm1webDojo",
                dijit: "tm1webDijit",
                dojox: "tm1webDojox",
                com: "tm1webCom"
            }
        }
    });

    require([
        "tm1web/api/Workbook"
    ], function(Workbook) {
        // Create and work with Workbook object
    });
</script>
</head>
<body class="claro tm1web"></body>
</html>
```

Loading Websheet objects with the JavaScript library

Use JavaScript to instantiate a Websheet object. After the object is loaded, you can then assign it as a descendant of the document body to display it in your web page.

You load a Websheet object by using the following format to specify the required properties and optional functions that define the object.

`new Workbook({properties ..., functions ...});`

The *properties* include values that specify the login credentials and the Websheet object that you want to open.

The *functions* can include optional code to notify you about `onLoad` and `onTitleDimensionElementChange` events for the object.

For more information, see "Cognos TM1 Web JavaScript library Workbook class" on page 179.

Example

The following example shows a JavaScript function that loads a Websheet object.

The code to instantiate the object must use the specific AMD (Asynchronous Module Definition) syntax and the AMD `require` keyword. After the object is created, the function assigns it as the child of a document body.

```
// Load Websheet with parameters for adminHost, tm1Server, username and password
function loadWebsheet() {
    require([
        "tm1web/api/Workbook"
    ], function(Workbook){
        var loadedWebsheet = new Workbook({
            adminHost: "localhost",
            tm1Server: "Planning Sample",
            username: "admin",
            password: "apple",
            path: "Applications/Planning Sample/Management Reporting/Actual v Budget",
            onLoad: function() {
                console.debug("Workbook loaded successfully.");
            }
        });

        // Add websheet to the document body
        document.body.appendChild(loadedWebsheet.domNode);

        loadedWebsheet.startup();
    });
};
```

The following example loads a Websheet object by using a session token for the login.

```
// Load Websheet with a session token
function loadWebsheet() {
    require([
        "tm1web/api/Workbook"
    ], function(Workbook){
        var loadedWebsheet = new Workbook({
            sessionToken: "yourSessionToken",
            path: "Applications/Planning Sample/Management Reporting/Actual v Budget",
            onLoad: function() {
                console.debug("Workbook loaded successfully.");
            }
        });
        // Add websheet to the document body
        document.body.appendChild(loadedWebsheet.domNode);

        loadedWebsheet.startup();
    });
};
```

Loading CubeViewer objects with the JavaScript library

Use JavaScript to instantiate a CubeViewer object. After the object is created, you can then assign it as a descendant of the document body to display it in your web page.

You load a CubeViewer object by using the following format to specify the required properties and optional functions that define the object.

`new` **`CubeViewer`**`(`*`{properties ..., functions ...}`*`);`

The *properties* include values that specify the login credentials and the CubeViewer object that you want to open.

The *functions* can include optional code to notify you about `onLoad` and `onTitleDimensionElementChange` events for the object.

For more information, see "Cognos TM1 Web JavaScript library CubeViewer class" on page 186.

Example

The following example shows a JavaScript function that loads a CubeViewer object.

The code to instantiate the object must use the specific AMD syntax and the Dojo `require` keyword. After the object is created, the function assigns it as the child of a document body.

```
function loadCubeview() {
    require([
        "tm1web/api/CubeViewer",
    ], function(CubeViewer) {
        var loadedCubeview = new CubeViewer({
            adminHost: "localhost",
            tm1Server: "Planning Sample",
            cube: "plan_BudgetPlan",
            view: "Budget Input Detailed",
            isPublic: true,
            onLoad: function() {
                console.debug("CubeViewer loaded successfully.");
            }
        });

        // Add cubeview to the document body
        document.body.appendChild(loadedCubeview.domNode);

        loadedCubeview.startup();
    });
};
```

The following example loads a CubeViewer object by using a session token for the login.

```
function loadCubeview() {
    require([
        "tm1web/api/CubeViewer",
    ], function(CubeViewer) {
        var loadedCubeview = new CubeViewer({
            sessionToken: "yourSessionToken",
            cube: "plan_BudgetPlan",
            view: "Budget Input Detailed",
            isPublic: true,
            onLoad: function() {
                console.debug("CubeViewer loaded successfully.");
            }
        });

        // Add cubeview to the document body
        document.body.appendChild(loadedCubeview.domNode);
        loadedCubeview.startup();
    });
};
```

JavaScript library callback functions

You can define a callback function when you instantiate Websheet and CubeViewer objects. The callback function traps changes to the title dimensions in the related object so you can process the event.

Websheet and CubeViewer objects both use the same format for defining a callback function. You add the callback function directly within the function that instantiates the TM1 Web object. Your code to handle the event goes inside this function.

Format

The callback function is defined with the following format:

```
onTitleDimensionElementChange: function(elementInfo) {

    // Add your code here to handle the title change event
}
```

When a change to a title dimension is detected, the `elementInfo` object is passed to the callback function. The content of `elementInfo` is different for Websheet and CubeViewer objects. Use this information to see which dimension title has changed.

Websheet elementInfo object:

sheetIndex
Type: Integer
The zero-based index of the sheet that contains the SUBNM cell that was changed.

rowIndex
Type: Integer
The zero-based index of the row that contains the SUBNM cell that was changed.

columnIndex
Type: Integer
The zero-based index of the column that contains the SUBNM cell that was changed.

dimension
Type: String
The name of the dimension.

element
Type: String
The name of the element.

elementIndex
Type: Integer
The one-based index of the dimension element.

CubeViewer elementInfo object:

dimension
Type: String
The name of the dimension.

element
Type: String
The name of the element.

elementIndex
Type: Integer
The one-based index of the dimension element.

Websheet callback function example

The following example shows a callback function that is defined within the function that loads a Websheet object.

```
function loadWebsheet() {
        require([
            "tm1web/api/Workbook"
        ], function(Workbook){
            var loadedWebsheet = new Workbook({
                sessionToken: "yourSessionToken",
                path: "Applications/Planning Sample/Management Reporting/Actual v 
Budget",,
                onLoad: function() {
                    console.debug("Workbook loaded successfully.");
                },
                onTitleDimensionElementChange: function(elementInfo) {
                    console.debug("Title dimension element changed:");
                    console.debug(elementInfo);
                }
            });

            document.body.appendChild(loadedWebsheet.domNode);

            loadedWebsheet.startup();
        });
    };
```

CubeViewer callback function example

The following example shows a callback function that is defined within the function that loads a CubeViewer object.

```
function loadCubeview() {
        require([
            "tm1web/api/CubeViewer"
        ], function(CubeViewer) {
            var loadedCubeview = new CubeViewer({
                sessionToken: "yourSessionToken",
                cube: "plan_BudgetPlan",
                view: "Budget Input Detailed",
                isPublic: true,
                onLoad: function() {
                    console.debug("CubeViewer loaded successfully.");
                },
                onTitleDimensionElementChange: function(elementInfo) {
                    console.debug("Title dimension element changed:");
                    console.debug(elementInfo);
                }
            });
            document.body.appendChild(loadedCubeview.domNode);

            loadedCubeview.startup();
        });
    };
```

JavaScript library sample code for properties and methods

After you load Websheet and CubeViewer objects with the Cognos TM1 Web JavaScript library, you can then apply the available properties and methods to them by using an object oriented approach.

The following code samples show how to apply different properties and methods.

Websheet object

- Rebuild the Active Forms in a Websheet
- Recalculate a Websheet

CubeViewer object

- Turn on/off auto recalculation mode
- Turn on/off the dimension title bar
- Reset a CubeViewer object to its original view
- Save a view
- Set the display mode and chart type

Websheet and CubeViewer objects

- Close a Websheet or CubeViewer object
- Logout

Example

```
<script type="text/javascript">

    // Rebuild the active form in a Websheet
    // ----------------------
    function rebuildActiveForms() {
        loadedWebsheet.rebuildActiveForms().then(
            function() {
                console.debug("Active form rebuild completed.");
            },
            function(message) {
                console.error(message);
            }
        );
    };


    // Recalculate a Websheet
    // ----------------------
    function recalculate() {
        loadedWebsheet.recalculate().then(
            function() {
                console.debug("Recalculate completed successfully.");
            },
            function(message) {
                console.error(message);
            }
        );
    };

    // Set the AutoRecalcMode for a CubeViewer object
    // ----------------------
    function toggleAutoRecalcMode(enabled) {
          loadedCubeview.set("automaticRecalculation", enabled).then(
            function() {
                var message = enabled ?
                    "Enabling auto recalc completed successfully." :
                    "Disabling auto recalc completed successfully.";
                console.debug(message);
            },
            function(message) {
                   console.error(message);
            }
        );
    };

    // Turn on/off the dimension title bar for a CubeViewer object
    // ----------------------
    function toggleDimensionBar(visible) {
          loadedCubeview.set("dimensionBarVisible", visible);
    };

    // Reset a CubeViewer object to it's original view
    // ----------------------
    function resetView() {
        loadedCubeview.reset().then(
            function() {
                console.debug("View reset completed successfully.");
            },
            function(message) {
                console.error(message);
            }
        );
    };

    // Save a view for a CubeViewer object
    // ----------------------
    function saveView() {
        loadedCubeview.save().then(
            function() {
                console.debug("Saving view completed successfully.");
            },
            function(message) {
                console.error(message);
            }
        );
    };

    // Close a Websheet or CubeViewer object
    // ----------------------
    function close() {
        loadedWebsheet.destroy();
    };

    // Set the display mode for a CubeViewer object
    // Valid values include Grid, Chart, GridAndChart
    // ----------------------
    function setDisplayMode() {

        require(["tm1web/cubeview/DisplayMode"], function(DisplayMode) {
            loadedCubeview.set("displayMode", DisplayMode.Grid).then(
                function() {
                    console.debug("Display mode change completed successfully.");
                },
                function(message) {
                    console.error(message);
                }
            );
        });
    };

    // Set the chart type for a CubeViewer object
    // ----------------------
    function setChartType() {

        require(["tm1web/cubeview/ChartType"], function(ChartType) {
            loadedCubeview.set("chartType", ChartType.Pie).then(
                function() {
                console.debug("Chart type change completed successfully.");
                },
                function(message) {
                console.error(message);
                }
            );
        });
    };
    // Logout from the session associated with the specified TM1 Web object
    // ---------------------
    function logout() {
        loadedCubeview.logout().then(
            function() {
                console.debug("Session destroyed.");
            },
            function(message) {
                console.error(message);
            }
        );
    };

</script>
```

Cognos TM1 Web JavaScript library Workbook class

The Workbook class represents a Cognos TM1 Web Websheet object.

Workbook objects extend the Dojo widget object (`dijit._WidgetBase`) and can be assigned as a child object of a Dojo tab container (`dijit.layout.TabContainer`) or other container. For more information, see Dojo documentation (http://dojotoolkit.org/documentation/).

In addition to the available properties and methods of the Dojo widget object, Workbook objects also have TM1 related properties and methods that you can access programmatically.

Workbook objects are loaded asynchronously and must finish loading before your code can interact with the objects.

Format

You load a Websheet object by using the following format to specify the required properties and optional functions that define the object.

`new` **`Workbook`**`({`*`properties ..., functions ...`*`});`

Properties

The *properties* include the following values that define the Websheet object.

- `adminHost`
- `tm1Server`
- `username`
- `password`
- `camPassport`
- `sessionToken`
- `objectId`
- `path`

Note: You can provide login credentials as either a session token and an object ID, or by including separate values for TM1 Admin host, TM1 Server, user name, password, or camPassport.

Functions

The *functions* can include the following optional code:

- Use the `onLoad` function so that you can be notified when the object is loaded and ready to interact with.
- Use the `onTitleDimensionElementChange` declaration so that you can process the event when a user changes a dimension title in the related object.
- Use the `OnActionButtonExecution` declaration so that you can process the event when an action button is executed.

Example

The following example shows a JavaScript function that loads a Websheet object.

The login credentials can be provided by using a session token.

Note: The Workbook class accepts `objectId` as a parameter during construction. The `objectId` should be included with a `sessionToken` to identify the TM1 Web session.

```
// Load Websheet with a session token
function loadWebsheet() {
    require([
        "tm1web/api/Workbook"
    ], function(Workbook){
        var loadedWebsheet = new Workbook({
            sessionToken: "yourSessionToken",
            objectId: "objectIdOfNewWorkbook"
            onLoad: function() {
                console.debug("Workbook loaded successfully.");
            }
        });

        // Add websheet to the document body
        document.body.appendChild(loadedWebsheet.domNode);

        loadedWebsheet.startup();

    });
};
```

Workbook properties

This Workbook class has the following properties.

When instantiating either a CubeViewer or Workbook, the following properties are common between the two objects.

sessionToken

Type: String

Specifies the TM1 Web session to use for this object. Do not use this property with the properties for `adminHost`, `tm1Server`, `username`, `password`, and `camPassport`. If this property is not specified, and no additional credentials are provided, the user is prompted with a login dialog during startup.

objectId

Type: String

The ID of the Workbook. A unique identifier that you can use to reference the specific Workbook.

The `objectId` should be included with a `sessionToken` to identify the TM1 Web session.

For example:

```
new Workbook({
    sessionToken: "previousSessionToken",
    objectId: "objectIdOfNewWorkbook"
});
```

adminHost

Type: String

Default: `localhost`

The admin host to use when the object is loaded. Do not use this property with the `sessionToken` property.

tm1Server

Type: String

The TM1 server to use when the object is loaded. Do not use this property with the `sessionToken` property. If unspecified and no `sessionToken` is provided, the user is prompted with a login dialog during startup.

username

Type: String

The user name to use when the object is loaded. Do not use this property with the `sessionToken` or `camPassport` properties. If unspecified and no `sessionToken` or `camPassport` is provided, the user is prompted with a login dialog during startup.

password
Type: String
The password to use when the object is loaded. If unspecified and no `sessionToken` is provided, the user is prompted with a login dialog during startup.

camPassport
Type: String
The Cognos BI authentication passport (CAM passport) to use when you load an object. Do not use this property with `username` or `sessionToken`.

domNode
Type: HTMLElement
The underlying HTML element that represents the widget. This property is automatically defined during object construction and should not be provided during instantiation.
For more information, see Dojo documentation for dijit._WidgetBase (https://dojotoolkit.org/reference-guide/1.10/dijit/_WidgetBase.html).

The following properties are used when you instantiate a Workbook object only.

path
Type: String
The path in the TM1 server application folder tree for the workbook to be loaded.
For example: `"Applications/Planning Sample/Bottom Up Input/Budget Input"`

replaceOnNavigate
Type: Boolean (default true)
If `true`, during action button navigation to a new workbook, this widget will be replaced with the new workbook and the existing workbook will be closed.
If `false`, it is the consumer's responsibility to create a new workbook or replace this one using the information provided to the `onActionButtonExecution` method.

Get properties

All properties that get a value are called with the following format:

`get("`*`property_Name`*`").`

For example: `get("sandboxes");`

sandboxes
Retrieves all available sandboxes.
Returns `dojo.promise.Promise` as a promise that is resolved when the sandboxes are retrieved. When the promise is resolved, an Array of objects that represent the available sandboxes is passed to any callback registered with the promise.
Each object uses the following format:

name
(String) - The name of the sandbox.
`active`
(Boolean) - `True` if this sandbox is the active sandbox for the object, else `false`.
baseSandbox
(Boolean) - `True` if this sandbox is the base sandbox, else `false`.
`defaultSandbox`
(Boolean) - `True` if this sandbox is the default sandbox, else `false`.

Set properties

All properties that set a value are called with the following format:

`set("`*`property_Name`*`", `*`value`*`)`

For example: `set("activeSandbox", "theSandbox");`

activeSandbox
Sets the specified sandbox as active.

Parameter: (String) *sandbox*. The name of the sandbox to set as active.
Returns: `dojo.promise.Promise` as a promise that is resolved when the active sandbox is set.

subset

Sets a subset object.
Parameter: (Object) *subset* An object that represents the dimension subset object to set. The object uses the following format:

sheetIndex
Type: Integer
The zero-based index of the sheet that contains the SUBNM cell whose dimension subset you want to change.

rowIndex
Type: Integer
The zero-based index of the row that contains the SUBNM cell whose dimension subset you want to change.

columnIndex
Type: Integer
The zero-based index of the column that contains the SUBNM cell whose dimension subset you want to change.

dimension
Type: String
The dimension name. Should not be used in conjunction with sheetIndex, rowIndex, and columnIndex.

setExpression
Type: String
The MDX expression used to define the subset. Not to be used in conjunction with subset. That is, either a setExpression or a subset name is provided from the input.

subset
Type: String
The subset name of the dimension subset to set. Not to be used in conjunction with setExpression.

alias
Type: String
The alias of the dimension subset to set.

element
Type: String
The name of the element. Not to be used with elementIndex.

elementIndex
Type: Integer
The one-based index of the dimension element to set. Not to be used with element.

Returns `dojo.promise.Promise` as a promise that is resolved when the subset objects are set. Any callbacks that are registered with the promise are passed an object that matches the format of the subset that is passed into this method. A value of null is passed if the subset was not changed.

subsets

Sets multiple subset objects.
Parameter: (Object[]) *subsets* An array of subset objects to set. Each object uses following format:

sheetIndex
Type: Integer
The zero-based index of the sheet that contains the SUBNM cell whose dimension subset you want to change.

rowIndex
Type: Integer
The zero-based index of the row that contains the SUBNM cell whose dimension subset you want to change.

columnIndex
Type: Integer
The zero-based index of the column that contains the SUBNM cell whose dimension subset you want to change.

dimension
Type: String
The dimension name. Should not be used in conjunction with sheetIndex, rowIndex, and columnIndex.

setExpression
Type: String
The MDX expression used to define the subset. Not to be used in conjunction with subset. That is, either a setExpression or a subset is provided from the input.

subset
Type: String
The subset name of the dimension subset to set. Not to be used in conjunction with setExpression.

alias
Type: String
The alias of the dimension subset to set.

element
Type: String
The name of the element. Not to be used with elementIndex.

elementIndex
Type: Integer
The one-based index of the dimension element to set. Not to be used with element.

Returns `dojo.promise.Promise` as a promise that is resolved when the subset objects are set. Any callbacks that are registered with the promise are passed an array of objects that match the format of the subset objects that are passed into this method for the subsets that are successfully set.

titleDimensionElement
Sets a title dimension element.
Parameter: (Object) *element* An object that represents the title dimension elements to set. The object uses the following format:

sheetIndex
Type: Integer
The zero-based index of the sheet that contains the SUBNM cell whose dimension element you want to change.

rowIndex
Type: Integer
The zero-based index of the row that contains the SUBNM cell whose dimension element you want to change.

columnIndex
Type: Integer
The zero-based index of the column that contains the SUBNM cell whose dimension element you want to change.

element
Type: String
The name of the element. Not to be used with `elementIndex`.

elementIndex
Type: Integer
The one-based index of the dimension element to set. Not to be used with `element`.

Returns `dojo.promise.Promise` as a promise that is resolved when the title dimension element is set. Any callbacks that are registered with the promise are passed an object that matches the format of the element that is passed into this method. A value of null is passed if the element was not changed.

titleDimensionElements
Sets multiple title dimension elements.
Parameter: (Object[]) *elements* An array of the title dimension elements to set. Each object uses following format:

sheetIndex
Type: Integer
The zero-based index of the sheet that contains the SUBNM cell for the dimension element that you want to change. Optional when used with `dimension`, but required for `rowIndex` and `columnIndex`.

rowIndex
Type: Integer
The zero-based index of the row that contains the SUBNM cell for the dimension element that you want to change. Do not use this parameter with the `dimension` parameter.

columnIndex
Type: Integer
The zero-based index of the column that contains the SUBNM cell for the dimension element that you want to change. Do not use this parameter with the `dimension` parameter.

dimension
Type: String

The name of the dimension. Do not use this parameter with rowIndex and columnIndex.

element

Type: String

The name of the element. Not to be used with elementIndex.

elementIndex

Type: Integer

The one-based index of the dimension element to set. Not to be used with element.

Returns dojo.promise.Promise as a promise that is resolved when the title dimension elements are set. Any callbacks that are registered with the promise are passed an array of objects that match the format of the element objects that are passed into this method for the elements that are successfully set.

Workbook methods

The Workbook class has the following methods.

startup

Begins the startup sequence for this object. Call this function after the object is added to the document. The onLoad method is run after the startup sequence completes.

Applies to both CubeViewer and Workbook objects.

Syntax: startup()

Example:

```
document.body.appendChild(loadedWebsheet.domNode);
loadedWebsheet.startup();
```

See the Dojo documentation for dijit._WidgetBase#startup.

commitActiveSandbox

Commits changed data in the active sandbox to the base sandbox.

Returns dojo.promise.Promise. A promise that is resolved when the sandbox commit attempt completes. Any callbacks that are registered with the promise are passed a boolean with a value of true if the sandbox commit was successful or a value of false if the commit was not successful.

copy

Copies the selected cells to the clipboard if a selection exists.

destroy

Destroys this object and prepares it for garbage collection.

See Dojo documentation for dijit._WidgetBase#destroy.

logout

Destroys the TM1 Web session that is associated with this object's sessionToken.

Returns dojo.promise.Promise as a promise that is resolved when the logout completes.

onActionButtonExecution

Called when an action button is executed.

Syntax: onActionButtonExecution: function(*executionResults*){}

Parameters: executionResults object that uses the following format.

calculation

Type: String

What type of calculation occurred on the current Workbook before action button execution occurred.

Value will be one of "None", "Recalculate", "Rebuild".

navigation

Type: Object

This property exists only if workbook or sheet navigation occurred as part of the action button execution.

calculation

Type: String

What type of calculation occurred on the target Workbook after the action button navigation occurred.

Value will be one of "None", "Recalculate", "Rebuild".

objectId

Type: String

The objectId of the workbook that has been navigated to. If an action on a worksheet within the same workbook occurred, the objectId will match the current workbook.

path

Type: String

The path to the workbook that was navigated to.

name

Type: String

The name of the target workbook.

sheetIndex

Type: Integer

The zero-based index of the worksheet that was navigated to.

replace

Type: Boolean

Whether the action button was configured to replace the existing workbook.

tiProcess

Type: Object

This property exists only if a TI process was executed as part of the action button's execution.

calculation

Type: String

What type of calculation occurred on the current workbook after the TI process was executed. Value will be one of "None, "Recalculate", "Rebuild".

name

Type: String

The name of the TI process that was executed.

executionSucceeded

Type: Boolean

Whether or not the TI process execution was successful.

onLoad

Runs when the object is finished loading.

onTitleDimensionElementChange

Executed when a title dimension element is changed. Can be overridden during object construction or attached to using dojo/aspect module.

Syntax: `onTitleDimensionElementChange: function(`*`elementInfo`*`){}`

Parameters: `elementInfo` object that uses the following format.

sheetIndex

Type: Integer

The zero-based index of the sheet that contains the SUBNM cell that was changed.

rowIndex

Type: Integer

The zero-based index of the row that contains the SUBNM cell that was changed.

columnIndex

Type: Integer

The zero-based index of the column that contains the SUBNM cell that was changed.

dimension

Type: String

The name of the dimension.

element

Type: String

The name of the element.

elementIndex

Type: Integer

The one-based index of the dimension element.

paste

Pastes the contents of the clipboard into the current selected area if a selection exists.

rebuildActiveForms

Rebuilds the active forms in the workbook.

Returns `dojo.promise.Promise` as a promise that is resolved when active forms are rebuilt.

redo

Performs a redo action.

Returns `dojo.promise.Promise` as a promise that is resolved when the redo action completes.

replace

Accepts an `objectId` and replaces the existing workbook with the one represented from the given `objectId` (unless it is the same as the existing websheet, in which case nothing no action is taken).

Replace assumes that the workbook that is replacing the existing one uses the same TM1 Web session as the previous workbook.

undo

Performs an undo action.

Returns `dojo.promise.Promise` as a promise that is resolved when the undo action completes.

Cognos TM1 Web JavaScript library CubeViewer class

The CubeViewer class represents a Cognos TM1 Web CubeViewer object.

CubeViewer objects extend the Dojo widget object (`dijit._WidgetBase`) and can be assigned as a child object of a Dojo tab container (`dijit.layout.TabContainer`) or other container. For more information, see Dojo documentation (http://dojotoolkit.org/documentation/).

In addition to the available properties and methods of the Dojo widget object, CubeViewer objects also have TM1 related properties and methods that you can access programmatically.

CubeViewer objects are loaded asynchronously and must finish loading before your code can interact with the objects.

Format

You load a CubeViewer object by using the following format to specify the required properties and optional functions that define the object.

```
new CubeViewer({properties ..., functions ...});
```

Properties

The *properties* include the following values that define the CubeViewer object.

- `adminHost`
- `tm1Server`
- `username`
- `password`
- `camPassport`
- `sessionToken`
- `objectId`
- `view`
- `cube`
- `isPublic`

Note: You can provide login credentials as either a session token and object ID, or by including separate values for TM1 Admin host, TM1 Server, user name, password, or camPassport.

Functions

The *functions* can include the following optional code:

- Use the `onLoad` function so that you can be notified when the object is loaded and ready to interact with.
- Use the `onTitleDimensionElementChange` declaration so you can process the event when a user changes a dimension title in the related object.

Example

The following example shows a JavaScript function that loads a CubeViewer object.

The login credentials are provided by using a session token.

```
function loadCubeview() {
    require([
        "tm1web/api/CubeViewer"
    ], function(CubeViewer) {
        var loadedCubeview = new CubeViewer({
            sessionToken: "yourSessionToken",
            cube: "plan_BudgetPlan",
            view: "Budget Input Detailed",
            isPublic: true,
            onLoad: function() {
                console.debug("CubeViewer loaded successfully.");
            },
        });

        // Add cubeview to the document body
        document.body.appendChild(loadedCubeview.domNode);

        loadedCubeview.startup();

    });
};
```

CubeViewer properties

The CubeViewer class has the following properties.

When instantiating either a CubeViewer or Workbook object, the following properties are common between the two types of objects:

sessionToken

Type: String
Specifies the TM1 Web session to use for this object. Do not use this property with the properties for `adminHost`, `tm1Server`, `username`, `password`, and `camPassport`. If this property is not specified, and no additional credentials are provided, the user is prompted with a login dialog during startup.

objectId

Type: String
The ID of the CubeViewer. A unique number that you can use to reference the specific CubeViewer.

adminHost

Type: String
Default: `localhost`
The admin host to use when the object is loaded. Do not use this property with the `sessionToken` property.

tm1Server

Type: String
The TM1 server to use when the object is loaded. Do not use this property with the `sessionToken` property. If unspecified and no `sessionToken` is provided, the user is prompted with a login dialog during startup.

username

Type: String
The user name to use when the object is loaded. Do not use this property with the `sessionToken` or `camPassport` properties. If unspecified and no `sessionToken` or `camPassport` is provided, the user is prompted with a login dialog during startup.

password

Type: String
The password to use when the object is loaded. If unspecified and no `sessionToken` is provided, the user is prompted with a login dialog during startup.

camPassport

Type: String
The Cognos BI authentication passport (CAM passport) to use when you load an object. Do not use this property with `username` or `sessionToken`.

domNode

Type: HTMLElement

The underlying HTML element that represents the widget. This property is automatically defined during object construction and should not be provided during instantiation.

For more information, see Dojo documentation for dijit._WidgetBase (https://dojotoolkit.org/reference-guide/1.10/dijit/_WidgetBase.html).

The following properties are used when you instantiate a CubeViewer object only.

view

Type: String

The name of the cube view to load.

cube

Type: String

The name of the cube that contains the view you want to load.

isPublic

Type: Boolean

Default: `true`

The access type of the cube view to load.

A value of `true` indicates that you want to load a public cube view.

A value of `false` indicates that you want to load a private cube view.

Get properties

All properties that get a value are called with the following format:

`get("`*`property_Name`*`").`

For example: `get("sandboxes");`

sandboxes

Retrieves all available sandboxes.

Returns `dojo.promise.Promise` as a promise that is resolved when the sandboxes are retrieved. When the promise is resolved, an Array of objects that represent the available sandboxes is passed to any callback registered with the promise.

Each object uses the following format:

- `name`: (String) - The name of the sandbox.
- `active`: (Boolean) - True if this sandbox is the active sandbox for the object, else false.
- `baseSandbox`: (Boolean) - True if this sandbox is the base sandbox, else false.
- `defaultSandbox`: (Boolean) - True if this sandbox is the default sandbox, else false.

Set properties

All properties that set a value are called with the following format:

`set("`*`property_Name`*`", `*`value`*`)`

For example: `set("activeSandbox", "theSandbox");`

activeSandbox

Sets the specified sandbox as active.

Parameter: (String) *sandbox*. The name of the sandbox to set as active.

Returns: `dojo.promise.Promise` as a promise that is resolved when the active sandbox is set.

automaticRecalculation

Sets automatic recalculation to on or off.

Parameters: Boolean.

- `True` turns on automatic recalculation.
- `False` turns off automatic recalculation.

Returns: `dojo.promise.Promise`. A promise that is resolved when automatic recalculation is enabled or disabled.

chartType
Sets the chart type of the CubeViewer object.
Parameters: `tm1web.cubeview.ChartType`. The chart type to set.
Returns: `dojo.promise.Promise`. A promise that is resolved when the chart type is set.

dimensionBarVisible
Sets the visibility of the dimension bar.
Parameters: Boolean.

- `True` turns on the display of the dimension bar.
- `False` turns off the display of the dimension bar.

displayMode
Sets the display mode of the CubeViewer object.
Parameters: `tm1web.cubeview.DisplayMode`. The display mode to set.
Returns: `dojo.promise.Promise`. A promise that is resolved when the display mode is set.

subset
Sets a subset object.
Parameter: (Object) *subset* An object that represents the dimension subset object to set. The object uses the following format:

dimension
Type: String
The dimension name.
setExpression
Type: String
The MDX expression used to define the subset. Not to be used in conjunction with subset. That is, either a setExpression or a subset name is provided from the input.
subset
Type: String
The subset name of the dimension subset to set. Not to be used in conjunction with setExpression.
alias
Type: String
The alias of the dimension subset to set.
element
Type: String
The name of the element. Not to be used with elementIndex.
elementIndex
Type: Integer
The one-based index of the dimension element to set. Not to be used with element.

Returns `dojo.promise.Promise` as a promise that is resolved when the subset objects are set. Any callbacks that are registered with the promise are passed an object that matches the format of the subset that is passed into this method. A value of null is passed if the subset was not changed.

subsets
Sets multiple subset objects.
Parameter: (Object[]) *subsets* An array of subset objects to set. Each object uses following format:

dimension
Type: String
The dimension name.
setExpression
Type: String
The MDX expression used to define the subset. Not to be used in conjunction with subset. That is, either a setExpression or a subset is provided from the input.
subset
Type: String
The subset name of the dimension subset to set. Not to be used in conjunction with setExpression.

alias
Type: String
The alias of the dimension subset to set.

element
Type: String
The name of the element. Not to be used with elementIndex.

elementIndex
Type: Integer
The one-based index of the dimension element to set. Not to be used with element.

Returns `dojo.promise.Promise` as a promise that is resolved when the subset objects are set. Any callbacks that are registered with the promise are passed an array of objects that match the format of the subset objects that are passed into this method for the subsets that are successfully set.

titleDimensionElement
Sets a title dimension element.
Parameter: element object. The title dimension element to set. This object uses the following format:

dimension
String
The name of the dimension.

element
String
The name of the element. Do not use this parameter with `elementIndex`.

elementIndex
Integer
The one-based index of the dimension element to set. Do not use this parameter with the `element` parameter.

Returns: `dojo.promise.Promise`. A promise that is resolved when the title dimension element is set. Any callbacks that are registered with the promise are passed an object that matches the format of the element that was passed into this method. A value of null is passed if the element was not changed.

titleDimensionElements
Sets multiple title dimension elements.
Parameter: object[] elements. An array of the title dimension elements to set. Each object uses the following format:

dimension
String
The name of the dimension.

element
String
The name of the element. Do not use this parameter with `elementIndex`.

elementIndex
Integer
The one-based index of the dimension element to set. Do not use this parameter with the `element` parameter.

Returns `dojo.promise.Promise`. A promise that is resolved when the title dimension elements are set. Any callbacks that are registered with the promise are passed an array of objects that match the format of the element objects that were passed into this method. The passed array reports back about the elements that are successfully set.

CubeViewer methods

The CubeViewer class has the following methods.

startup
Begins the startup sequence for this object. Call this function after the object is added to the document. The `onLoad` method is run after the startup sequence completes.
Applies to both CubeViewer and Workbook objects.
Syntax: `startup()`

Example:

```
document.body.appendChild(loadedCubeViewer.domNode);
loadedCubeViewer.startup();
```

See the Dojo documentation for `dijit._WidgetBase#startup`.

commitActiveSandbox

Commits changed data in the active sandbox to the base sandbox.

Returns `dojo.promise.Promise`. A promise that is resolved when the sandbox commit attempt is completed. Any callbacks that are registered with the promise are passed a boolean with a value of true if the sandbox commit was successful. A value of false is passed if the commit was unsuccessful.

copy

Copies the selected cells to the clipboard if a selection exists.

destroy

Destroys this object and prepares it for garbage collection.

See Dojo documentation for `dijit._WidgetBase#destroy`.

logout

Destroys the TM1 Web session that is associated with this object's sessionToken.
Returns `dojo.promise.Promise` as a promise that is resolved when the logout completes.

onLoad

Runs when the object is finished loading.

onTitleDimensionElementChange

Executed when a title dimension element is changed. Can be overridden during object construction or attached to by using the dojo/aspect module.
Syntax: `onTitleDimensionElementChange: function(`*`elementInfo`*`){}`
Parameter: `elementInfo` object. This object uses the following format:

dimension

String
The name of the dimension that was changed.

element

String
The name of the element that was changed.

elementIndex

Integer
The one-based index of the dimension element that was changed.

paste

Pastes the contents of the clipboard into the current selected area if a selection exists.

redo

Performs a redo action.
Returns `dojo.promise.Promise`) as a promise that is resolved when the redo action completes.

reset

Resets the cube view to its original saved state.

Returns: `dojo.promise.Promise`. A promise that is resolved when the cube view is reset.

save

Saves the layout of cube view and overwrites the existing layout.

Returns: `dojo.promise.Promise`. A promise that is resolved when the cube view is saved.

undo

Performs an undo action.
Returns `dojo.promise.Promise` as a promise that is resolved when the undo action completes.

Appendix A. Supported Microsoft Excel Functions - TM1 Web

IBM Cognos TM1 Web supports many Excel worksheet functions. This appendix lists the supported Excel functions by category and in alphabetical order, and describes any differences in performance between the Excel functions and TM1 Web functions.

Date and Time Functions

The following table lists date and time functions.

Function	**Description**
DATE	Returns the serial number of a particular date.
DATEVALUE	Converts a date in the form of text to a serial number.
DAY	Converts a serial number to a day of the month.
DAYS360	Calculates the number of days between two dates based on a 360-day year.
HOUR	Converts a serial number to an hour.
MINUTE	Converts a serial number to a minute.
MONTH	Converts a serial number to a month.
NOW	Returns the serial number of the current date and time.
SECOND	Converts a serial number to a second.
TIME	Returns the serial number of a particular time.
TIMEVALUE	Converts a time in the form of text to a serial number.
TODAY	Returns the serial number of today's date.
WEEKDAY	Converts a serial number to a day of the week.
YEAR	Converts a serial number to a year.

Financial Functions

The following table lists financial functions.

Function	**Description**
DB	Returns the depreciation of an asset for a specified period using the fixed-declining balance method.

© Copyright IBM Corp. 2007, 2017

Function	Description
DDB	Returns the depreciation of an asset for a specified period using the double-declining balance method or some other method you specify.
FV	Returns the future value of an investment.
IPMT	Returns the interest payment for an investment for a given period.
IRR	Returns the internal rate of return for a series of cash flows.
ISPMT	Calculates the interest paid during a specific period of an investment.
MIRR	Returns the internal rate of return where positive and negative cash flows are financed at different rates.
NPER	Returns the number of periods for an investment.
NPV	Returns the net present value of an investment based on a series of periodic cash flows and a discount rate.
PMT	Returns the periodic payment for an annuity.
PPMT	Returns the payment on the principal for an investment for a given period.
PV	Returns the present value of an investment.
RATE	Returns the interest rate per period of an annuity.
SLN	Returns the straight-line depreciation of an asset for one period.
SYD	Returns the sum-of-years' digits depreciation of an asset for a specified period.

Information Functions

The following table lists information functions that are supported in TM1 Web.

Function	Description
CELL	Returns information about the formatting, location, or contents of a cell. Support for the Cell function is limited to the following info_types: address, col, row, protect, contents, type.
ISERR	Returns TRUE if the value is any error value except #N/A.
ISERROR	Returns TRUE if the value is any error value.
ISNA	Returns TRUE if the value is the #N/A error value.
NA	Returns the error value #N/A.

Logical Functions

The following table lists logical functions.

Function	Description
AND	Returns TRUE if all its arguments are TRUE.
FALSE	Returns the logical value FALSE.
IF	Specifies a logical test to perform.
NOT	Reverses the logic of its argument.
OR	Returns TRUE if any argument is TRUE.
TRUE	Returns the logical value TRUE.

Lookup and Reference Functions

The following table lists lookup and reference functions.

Note: Certain functions, such as LOOKUP and ROWS, may accept two dimensional arrays as arguments. TM1 Web does not support two dimensional arrays. Depending on the data organization and requirements, these functions can still obtain correct values, for example, when the data being retrieved falls in the initial portions of the array. To ensure correct values when working with these functions on TM1 Web you may need to reorganize the input data into repeated functions using one dimensional arrays or you may need to use direct cell references.

Function	Description
ADDRESS	Returns a reference as text to a single cell in a worksheet.
CHOOSE	Chooses a value from a list of values.
COLUMN	Returns the column number of a reference.
COLUMNS	Returns the number of columns in a reference.
HLOOKUP	Looks in the top row of an array and returns the value of the indicated cell.
HYPERLINK	Creates a shortcut or jump that opens a document stored on a network server, an intranet, or the Internet.
INDEX	Uses an index to choose a value from a reference or array.
LOOKUP	Looks up values in a vector or array.
MATCH	Looks up values in a reference or array.
OFFSET	Returns a reference offset from a given reference.
ROW	Returns the row number of a reference.
ROWS	Returns the number of rows in a reference.

Function	Description
VLOOKUP	Looks in the first column of an array and moves across the row to return the value of a cell.

Math and Trigonometric Functions

The following table lists math and trigonometric functions.

Function	Description
ABS	Returns the absolute value of a number.
ACOS	Returns the arccosine of a number.
ACOSH	Returns the inverse hyperbolic cosine of a number.
ASIN	Returns the arcsine of a number.
ASINH	Returns the inverse hyperbolic sine of a number.
ATAN	Returns the arctangent of a number.
ATAN2	Returns the arctangent from x- and y-coordinates.
ATANH	Returns the inverse hyperbolic tangent of a number.
CEILING	Rounds a number to the nearest integer or to the nearest multiple of significance.
COMBIN	Returns the number of combinations for a given number of objects.
COS	Returns the cosine of a number.
COSH	Returns the hyperbolic cosine of a number.
DEGREES	Converts radians to degrees.
EVEN	Rounds a number up to the nearest even integer.
EXP	Returns e raised to the power of a given number.
FACT	Returns the factorial of a number.
FLOOR	Rounds a number down, toward zero.
INT	Rounds a number down to the nearest integer.
LN	Returns the natural logarithm of a number.
LOG	Returns the logarithm of a number to a specified base.
LOG10	Returns the base-10 logarithm of a number.
MOD	Returns the remainder from division.

Function	Description
ODD	Rounds a number up to the nearest odd integer.
PI	Returns the value of pi.
POWER	Returns the result of a number raised to a power.
PRODUCT	Multiplies its arguments.
RADIANS	Converts degrees to radians.
RAND	Returns a random number between 0 and 1.
ROMAN	Converts an arabic numeral to roman, as text.
ROUND	Rounds a number to a specified number of digits.
ROUNDDOWN	Rounds a number down, toward zero.
ROUNDUP	Rounds a number up, away from zero.
SIGN	Returns the sign of a number.
SIN	Returns the sine of the given angle.
SINH	Returns the hyperbolic sine of a number.
SQRT	Returns a positive square root.
SUM	Adds its arguments.
SUMIF	Adds the cells specified by a given criteria.
TAN	Returns the tangent of a number.
TANH	Returns the hyperbolic tangent of a number.

Text and Data Functions

The following table lists text and data functions.

Function	Description
CHAR	Returns the character specified by the code number.
CLEAN	Removes all nonprintable characters from text.
CODE	Returns a numeric code for the first character in a text string.
CONCATENATE	Joins several text items into one text item.
DOLLAR	Converts a number to text, using the $ (dollar) currency format.

Function	Description
EXACT	Checks to see if two text values are identical.
FIND	Finds one text value within another (case-sensitive).
FIXED	Formats a number as text with a fixed number of decimals.
LEFT	Returns the leftmost characters from a text value.
LEN	Returns the number of characters in a text string.
LOWER	Converts text to lowercase.
MID	Returns a specific number of characters from a text string starting at the position you specify.
PROPER	Capitalizes the first letter in each word of a text value.
REPLACE	Replaces characters within text.
REPT	Repeats text a given number of times.
RIGHT	Returns the rightmost characters from a text value.
SEARCH	Finds one text value within another (not case-sensitive).
SUBSTITUTE	Substitutes new text for old text in a text string.
T	Converts its arguments to text.
TEXT	Formats a number and converts it to text.
TRIM	Removes spaces from text.
UPPER	Converts text to uppercase.
VALUE	Converts a text argument to a number.

Statistical Functions

The following table lists statistical functions.

Function	Description
AVEDEV	Returns the average of the absolute deviations of data points from their mean.
AVERAGE	Returns the average of its arguments.
AVERAGEA	Returns the average of its arguments, including numbers, text, and logical values.
BINOMDIST	Returns the individual term binomial distribution probability.
CONFIDENCE	Returns the confidence interval for a population mean.

Function	Description
CORREL	Returns the correlation coefficient between two data sets.
COUNT	Counts how many numbers are in the list of arguments.
COUNTA	Counts how many values are in the list of arguments.
COUNTIF	Counts the number of nonblank cells within a range that meet the given criteria.
COVAR	Returns covariance, the average of the products of paired deviations.
DEVSQ	Returns the sum of squares of deviations.
EXPONDIST	Returns the exponential distribution.
FISHER	Returns the Fisher transformation.
FISHERINV	Returns the inverse of the Fisher transformation.
FORECAST	Returns a value along a linear trend.
GEOMEAN	Returns the geometric mean.
GROWTH	Returns values along an exponential trend.
HARMEAN	Returns the harmonic mean.
INTERCEPT	Returns the intercept of the linear regression line.
KURT	Returns the kurtosis of a data set.
LARGE	Returns the k-th largest value in a data set.
LINEST	Returns the parameters of a linear trend.
LOGEST	Returns the parameters of an exponential trend.
MAX	Returns the maximum value in a list of arguments.
MATCH	Returns the relative position of an item in an array that matches a specified value in a specified order.
MAXA	Returns the maximum value in a list of arguments, including numbers, text, and logical values.
MEDIAN	Returns the median of the given numbers.
MIN	Returns the minimum value in a list of arguments.
MINA	Returns the smallest value in a list of arguments, including numbers, text, and logical values.
NEGBINOMDIST	Returns the negative binomial distribution, the probability that there will be Number_f failures before the Number_s-th success, with Probability_f probability of a success.

Function	Description
MODE	Returns the most common value in a data set.
NORMDIST	Returns the normal cumulative distribution.
NORMINV	Returns the inverse of the normal cumulative distribution.
NORMSDIST	Returns the standard normal cumulative distribution.
NORMSINV	Returns the inverse of the standard normal cumulative distribution.
PEARSON	Returns the Pearson product moment correlation coefficient.
PERMUT	Returns the number of permutations for a given number of objects.
RSQ	Returns the square of the Pearson product moment correlation coefficient.
SKEW	Returns the skewness of a distribution.
SLOPE	Returns the slope of the linear regression line.
SMALL	Returns the k-th smallest value in a data set.
STANDARDIZE	Returns a normalized value.
STDEV	Estimates standard deviation based on a sample.
STDEVA	Estimates standard deviation based on a sample, including numbers, text, and logical values.
STDEVP	Calculates standard deviation based on the entire population.
STDEVPA	Calculates standard deviation based on the entire population, including numbers, text, and logical values.
STEYX	Returns the standard error of the predicted y-value for each x in the regression.
TREND	Returns values along a linear trend.
VAR	Estimates variance based on a sample.
VARA	Estimates variance based on a sample, including numbers, text, and logical values.
VARP	Calculates variance based on the entire population.
VARPA	Calculates variance based on the entire population, including numbers, text, and logical values.
WEIBULL	Returns the Weibull distribution.

Appendix B. Unsupported Microsoft Excel Functions - TM1 Web

IBM Cognos TM1 Web supports many Excel worksheet functions. This appendix lists the Excel functions, by category and in alphabetical order, that are not supported in TM1 Web.

Database and List Management Functions

This table lists the management functions that are not supported in TM1 Web.

Function	Description
DAVERAGE	Returns the average of selected database entries.
DCOUNT	Counts the cells that contain numbers in a database.
DCOUNTA	Counts nonblank cells in a database.
DGET	Extracts from a database a single record that matches the specified criteria.
DMAX	Returns the maximum value from selected database entries.
DMIN	Returns the minimum value from selected database entries.
DPRODUCT	Multiplies the values in a particular field of records that match the criteria in a database.
DSTDEV	Estimates the standard deviation based on a sample of selected database entries.
DSTDEVP	Calculates the standard deviation based on the entire population of selected database entries.
DSUM	Adds the numbers in the field column of records in the database that match the criteria.
DVAR	Estimates variance based on a sample from selected database entries.
DVARP	Calculates variance based on the entire population of selected database entries.

Date and Time Functions

This table lists the date and time functions that are not supported in TM1 Web.

Function	Description
EDATE	Returns the serial number of the date that is the indicated number of months before or after the start date.
EOMONTH	Returns the serial number of the last day of the month before or after a specified number of months.

© Copyright IBM Corp. 2007, 2017

Function	Description
NETWORKDAYS	Returns the number of whole workdays between two dates.
WEEKNUM	Converts a serial number to a number representing where the week falls numerically with a year.
WORKDAY	Returns the serial number of the date before or after a specified number of workdays.
YEARFRAC	Returns the year fraction representing the number of whole days between start_date and end_date.

Financial Functions

This table lists the financial functions that are not supported in TM1 Web.

Functions	Description
ACCRINT	Returns the accrued interest for a security that pays periodic interest.
ACCRINTM	Returns the accrued interest for a security that pays interest at maturity.
AMORDEGRC	Returns the depreciation for each accounting period by using a depreciation coefficient.
AMORLINC	Returns the depreciation for each accounting period.
COUPDAYBS	Returns the number of days from the beginning of the coupon period to the settlement date.
COUPDAYS	Returns the number of days in the coupon period that contains the settlement date.
COUPDAYSNC	Returns the number of days from the settlement date to the next coupon date.
COUPNCD	Returns the next coupon date after the settlement date.
COUPNUM	Returns the number of coupons payable between the settlement date and maturity date.
COUPPCD	Returns the previous coupon date before the settlement date.
CUMIPMT	Returns the cumulative interest paid between two periods.
CUMPRINC	Returns the cumulative principal paid on a loan between two periods.
DISC	Returns the discount rate for a security.
DOLLARDE	Converts a dollar price, expressed as a fraction, into a dollar price, expressed as a decimal number.
DOLLARFR	Converts a dollar price, expressed as a decimal number, into a dollar price, expressed as a fraction.
DURATION	Returns the annual duration of a security with periodic interest payments.
EFFECT	Returns the effective annual interest rate.

Functions	Description
FVSCHEDULE	Returns the future value of an initial principal after applying a series of compound interest rates.
INTRATE	Returns the interest rate for a fully invested security.
MDURATION	Returns the Macauley modified duration for a security with an assumed par value of $100.
NOMINAL	Returns the annual nominal interest rate.
ODDFPRICE	Returns the price per $100 face value of a security with an odd first period.
ODDFYIELD	Returns the yield of a security with an odd first period.
ODDLPRICE	Returns the price per $100 face value of a security with an odd last period.
ODDLYIELD	Returns the yield of a security with an odd last period.
PRICE	Returns the price per $100 face value of a security that pays periodic interest.
PRICEDISC	Returns the price per $100 face value of a discounted security.
PRICEMAT	Returns the price per $100 face value of a security that pays interest at maturity.
RECEIVED	Returns the amount received at maturity for a fully invested security.
TBILLEQ	Returns the bond-equivalent yield for a Treasury bill.
TBILLPRICE	Returns the price per $100 face value for a Treasury bill.
TBILLYIELD	Returns the yield for a Treasury bill.
VDB	Returns the depreciation of an asset for a specified or partial period using a declining balance method.
XIRR	Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic.
XNPV	Returns the net present value for a schedule of cash flows that is not necessarily periodic.
YIELD	Returns the yield on a security that pays periodic interest.
YIELDDISC	Returns the annual yield for a discounted security; for example, a Treasury bill.
YIELDMAT	Returns the annual yield of a security that pays interest at maturity.

Information Functions

This table lists the information functions that are not supported in TM1 Web.

Function	Description
ERROR.TYPE	Returns a number corresponding to an error type.

Function	Description
INFO	Returns information about the current operating environment.
ISBLANK	Returns TRUE if the value is blank.
ISEVEN	Returns TRUE if the number is even.
ISLOGICAL	Returns TRUE if the value is a logical value.
ISNONTEXT	Returns TRUE if the value is not text.
ISNUMBER	Returns TRUE if the value is a number.
ISODD	Returns TRUE if the number is odd.
ISREF	Returns TRUE if the value is a reference.
ISTEXT	Returns TRUE if the value is text.
N	Returns a value converted to a number.
TYPE	Returns a number indicating the data type of a value.

Lookup and Reference Functions

This table lists the lookup and reference functions that are not supported in TM1 Web.

Function	Description
AREAS	Returns the number of areas in a reference.
INDIRECT	Returns a reference indicated by a text value.
RTD	Retrieves real-time data from a program that supports COM automation.
TRANSPOSE	Returns the transpose of an array.

Math and Trigonometric Functions

This table lists the math and trigonometric functions that are not supported in TM1 Web.

Function	Description
FACTDOUBLE	Returns the double factorial of a number.
GCD	Returns the greatest common divisor.
LCM	Returns the least common multiple.
MDETERM	Returns the matrix determinant of an array.

Function	Description
MINVERSE	Returns the matrix inverse of an array.
MMULT	Returns the matrix product of two arrays.
MROUND	Returns a number rounded to the desired multiple.
MULTINOMIAL	Returns the multinomial of a set of numbers.
QUOTIENT	Returns the integer portion of a division.
RANDBETWEEN	Returns a random number between the numbers you specify.
SERIESSUM	Returns the sum of a power series based on the formula.
SQRTPI	Returns the square root of (number * pi).
SUBTOTAL	Returns a subtotal in a list or database.
SUMPRODUCT	Returns the sum of the products of corresponding array components.
SUMSQ	Returns the sum of the squares of the arguments.
SUMX2MY2	Returns the sum of the difference of squares of corresponding values in two arrays.
SUMX2PY2	Returns the sum of the sum of squares of corresponding values in two arrays.
SUMXMY2	Returns the sum of squares of differences of corresponding values in two arrays.
TRUNC	Truncates a number to an integer.

Statistical Functions

This table lists the statistical functions that are not supported in TM1 Web.

Function	Description
BETADIST	Returns the beta cumulative distribution function.
BETAINV	Returns the inverse of the cumulative distribution function for a specified beta distribution.
CHIDIST	Returns the one-tailed probability of the chi-squared distribution.
CHIINV	Returns the inverse of the one-tailed probability of the chi-squared distribution.
CHITEST	Returns the test for independence.
COUNTBLANK	Counts the number of blank cells within a range.
CRITBINOM	Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value.
FDIST	Returns the F probability distribution.

Function	Description
FINV	Returns the inverse of the F probability distribution.
FREQUENCY	Returns a frequency distribution as a vertical array.
FTEST	Returns the result of an F-test.
GAMMADIST	Returns the gamma distribution.
GAMMAINV	Returns the inverse of the gamma cumulative distribution.
GAMMALN	Returns the natural logarithm of the gamma function, G(x).
HYPGEOMDIST	Returns the hyper geometric distribution.
LOGINV	Returns the inverse of the lognormal distribution.
LOGNORMDIST	Returns the cumulative lognormal distribution.
NEGBINOMDIST	Returns the negative binomial distribution.
PERCENTILE	Returns the k-th percentile of values in a range.
PERCENTRANK	Returns the percentage rank of a value in a data set.
POISSON	Returns the Poisson distribution.
PROB	Returns the probability that values in a range are between two limits.
QUARTILE	Returns the quartile of a data set.
RANK	Returns the rank of a number in a list of numbers.
TDIST	Returns the Student's t-distribution.
TINV	Returns the inverse of the Student's t-distribution.
TRIMMEAN	Returns the mean of the interior of a data set.
TTEST	Returns the probability associated with a Student's t-test.
ZTEST	Returns the one-tailed probability-value of a z-test.

Text and Data Functions

This table lists the text and data functions that are not supported in TM1 Web.

Function	Description
ASC	Changes full-width (double-byte) English letters or katakana within a character string to half-width (single-byte) characters.
BAHTTEXT	Converts a number to text, using the ß (baht) currency format.

Function	Description
JIS	Changes half-width (single-byte) English letters or katakana within a character string to full-width (double-byte) characters.
PHONETIC	Extracts the phonetic (furigana) characters from a text string.
AutoShapes	TM1 Web does not support Microsoft Office Autoshapes.

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.

© Copyright IBM Corp. 2007, 2017

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 herein 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.

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 2.0.0 and may also apply to subsequent releases.

Copyright

Licensed Materials - Property of IBM

© Copyright IBM Corp. 2007, 2017.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- 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.

Microsoft product screen shot(s) used with permission from Microsoft.

Index