IBM Cognos TM1
Version 10.2.0

Reference Guide

Note

Before using this information and the product it supports, read the information in “Notices” on page 325.

Product Information

This document applies to IBM Cognos TM1 Version 10.2.0 and may also apply to subsequent releases.

Contents

Chapter 3. Rules Functions . . . 103

Introduction

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

This document is a collection of reference material for the IBM Cognos TM1 Business Analytics software functions, variables, and other programming elements.

Business Analytics provides software solutions for the continuous management and monitoring of Financial, Operational, Customer and Organizational performance across the enterprise.

Finding information

To find IBM Cognos product documentation on the web, including all translated documentation, access one of the IBM Cognos Information Centers (http://pic.dhe.ibm.com/infocenter/cogic/v1r0m0/index.jsp). Release Notes are published directly to Information Centers, and include links to the latest technotes and APARs.

Samples disclaimer

The Samples Outdoors Company, GO Sales, any variation of the Great Outdoors name, 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. What's new

This section contains a list of new, changed, and removed features for this release.

It will help you plan your upgrade and application deployment strategies and the training requirements for your users.

To locate the most current product documentation, go to the IBM Cognos TM1 information center (http://pic.dhe.ibm.com/infocenter/ctm1/v10r2m0/index.jsp)

New features in version 10.2.0

Listed below are new functions added to the documentation for IBM Cognos TM1 since the last release.

The following new functions were added to the TurboIntegrator Functions documentation:

- “ServerSandboxGet” on page 264
- “ServerSandboxExists” on page 263
- “ServerSandboxListCountGet” on page 264
- “AssociateCAMIDToGroup” on page 267
- “RemoveCAMIDAssociation” on page 271
- “RemoveCAMIDAssociationFromGroup” on page 271
- “CellSecurityCubeCreate” on page 267
- “CellSecurityCubeDestroy” on page 268

Chapter 2. Windows and Dialog Boxes

This section describes all significant IBM Cognos TM1 windows and dialog boxes.

Action Button Properties Dialog Box

Use the Action Button Properties dialog box to add TM1 Action buttons to a worksheet. You can configure the button to run a process and/or navigate to another worksheet.

For examples and steps on using Action buttons in worksheets, see the *IBM Cognos TM1 Developer Guide*.

Server

This list includes the names of all TM1 servers currently available on your network.

Select the server where the process or target worksheet is located for your Action button.

Connect

This button is available only when you are not connected to the server currently selected in the server list box.

Click this button to connect to the server that you selected in the server list box.

Disconnect

This button is available only when you are connected to the server currently selected in the server list box.

Click this button to disconnect from the server that you selected in the server box.

Action

Select the action that you want the Action button to perform when it is clicked.

- Run a TurboIntegrator Process

  Select this option to configure an Action button that runs a process. When you select this option, the Process tab becomes enabled.
- Go to another Worksheet

  Select this option to configure an Action button that navigates to another worksheet. When you select this option, the Worksheet tab becomes enabled.
- Run a Process, then go to a Worksheet

  Select this option to configure an Action button that runs a process and then navigates to another worksheet. When you select this option, both the Process and Worksheet tabs become enabled.
- Calculate/Rebuild Only

  Select this option to recalculate or rebuild without running a TI process or navigating to a new worksheet. This can be useful if you want to update only 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.

OK

Closes the Action Button Properties dialog box and inserts an Action button into your worksheet.

Cancel

Closes the Action Button Properties dialog box without inserting an Action button.

Process Tab

Use the Process tab to configure an Action button to run a process.

Process

Use this list to select the process you want to run in one of the following ways:

- To run a process that is available on the current server, select the process name from the list.
- To retrieve both the process name and parameter values from the current worksheet, select Get Process info from Worksheet.

Options

Opens the Process Options dialog where you can control the behavior of the Action button *before* and *after* the process is run.

For details, see the section “Process Options Dialog Box” on page 44.

Process Name

This option appears only when you select the Get Process info from Worksheet in the Process list.

Enter an Excel reference that provides the name of the process to run in one of the following ways.

- To reference a single cell, use the following format: =ColumnNameRowName. For example: =A1.
- To reference a named range in Excel, use the following format: =NameOfRange
- To select the cell from the current worksheet, click the **Excel Reference** button next to the Process Name box.

Parameters

Enter values for the process parameters, depending on how you selected the process name from the Process list.

- If you selected a process from the Process list, the Parameters grid appears with a list of the parameters for the selected process. You can enter values for each parameter directly into the grid or use an Excel reference that dynamically retrieves a parameter value from the current worksheet.
- If you selected the Get Process info from Worksheet option in the Process list, you must use an Excel reference to retrieve the parameter values from the current worksheet. You can enter a reference to a single cell, a range of cells, or a named range. Any reference must point to the appropriate number of cells, depending on the number of parameters that the process is expecting.

Click the **Excel Reference** button to directly select the cell or range of cells from the worksheet.

For examples, see the *IBM Cognos TM1 Developer Guide*.

Excel Reference

Creates an Excel reference that dynamically retrieves the process name or parameter value(s) from the current worksheet when the **Action** button is clicked.

Worksheet Tab

Use the Worksheet tab to configure an Action button to navigate to another Excel worksheet.

Look In

Use one of the following methods to select a worksheet:

- TM1 Applications - Select this option if you want to choose a worksheet from the TM1 Applications tree.
- Files - Select this option if you want to choose a worksheet from your computer.

Browse

Click this button to select the worksheet to which you want to navigate.

- If you selected the TM1 Applications option, a dialog box opens where you can select a worksheet from the TM1 Applications tree.
- If you selected the Files option, the Open dialog box appears where you can browse and select a file from your computer.

Workbook

Contains the path and name of the Excel workbook to which you want to navigate. You can enter this value in one of the following ways:

- Click the **Browse** button next to the Look In option to select a workbook from either the TM1 Applications tree or from the files on your computer.
- Click the **Excel Reference** button to select a cell that evaluates to a workbook path and name.
- Manually enter a workbook name and path.
- Manually enter an Excel reference that evaluates to a workbook path and name.

The path for a workbook in the TM1 Applications tree uses the format:

<FolderName>\<FolderName>\<WorkbookName>

For example:

Planning Sample\Bottom Up Input\Budget Input

The path for a network file uses the format:

\\<ComputerName>\<FolderName>\<WorkbookName>

For example:

\\boston\reports\2007_summary.xls

For details and examples, see the IBM Cognos TM1 *Developer Guide*.

Sheet

Contains the name of the worksheet to which you want to navigate. You can enter this value in one of the following ways:

- Click the **Browse** button to select a workbook and then select a worksheet from the Sheet list.
- Manually enter a worksheet name.
- Manually enter an Excel reference that evaluates to a worksheet name.
- Click the **Excel Reference** button to select a cell that evaluates to a worksheet name.

For details and examples, see the IBM Cognos TM1 *Developer Guide*.

Match Title Elements

This option automatically matches and sets the title dimensions between the source and target worksheets when a user clicks the **Action** button to navigate to the target worksheet.

For details and examples, see the IBM Cognos TM1 *Developer Guide*.

Replace Current Workbook

This option determines how the target worksheet is opened.

- If this option is not selected (default), the target worksheet is opened in a new window in Excel or on a new tab in TM1 Web .
- If this option is selected, the target worksheet is opened in the same window or tab, replacing the source worksheet.

CAUTION:
If you enable this 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.**

Advanced Options

Click this button to open the Advanced Options dialog box where you can manually map fields between the source and target worksheets for an Action button that navigates from one worksheet to another.

For details, see "Advanced Options Dialog Box" on page 7.

Appearance Tab

Use the Appearance tab to configure the visual appearance of the Action button.

Caption

Sets the caption text that displays on the Action button.

Font

Click this button to display the Font dialog box where you can set the font style and size for 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 Action button.

Select this option and then click **Browse** to locate and select the image file that you want to use.

Display as Hyperlink

Displays the Action button as a hyperlink with blue, underlined text instead of a standard button.

This option is not available when you select the Show Background Image option.

Preview

This area shows a preview of the text caption, font style, font color and background color for the button.

Colors

Allows you to set the text and background colors for the Action button.

Click the Text or Background color sample to display the Color dialog box where you can select a standard color or define a custom color.

This option is not available when you select the Display as Hyperlink option.

Advanced Options Dialog Box

Use the Advanced Options dialog box to manually map fields between the source and target worksheets when you insert 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.

Field	Description
Add	Adds a new row to the Advanced Mapping grid.
Delete	Deletes the selected row from the Advanced Mapping grid.
OK	Closes the Advanced Options dialog box and saves your settings.
Cancel	Closes the Advanced Options dialog box without saving your settings.

For examples on using the Advanced Options dialog box, see the IBM Cognos TM1 *Developer Guide*.

Advanced Mapping Grid

Use the Advanced Mapping grid to define the mapping of fields between the source and target worksheets. 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.

Field	Description
Source Type	This 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.
Source Object	This field takes 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. You can also retrieve these values from the source worksheet by using the = symbol to create an Excel reference.

Field	Description
Target Type	This field 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. **CAUTION:** **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 navigate with the Action button. If the target cell contains a TM1 DBRW function, then the function will be lost and the cell will not be able to connect to, read from, or write to the server.**
Target Object	This field represents the location in the target worksheet where the value from the Source Object will be inserted. Enter the Target Object as follows, depending on your selection for Target Type: • If Target Type is set to SUBNM, you need to specify the name of the title dimension in the target worksheet. • If Target Type is set to Named Range, you need to specify the name of the range in the target worksheet. • If Target Type is set to Range, you need to specify the cell location in the target worksheet. You can also use an Excel reference to retrieve the value for the Target Object field. For a detailed example, see the IBM Cognos TM1 *Developer Guide*.
Subset	Enter a value for the Subset field when the Target Type field is set to SUBNM.
Alias	Enter a value for the Alias field when the Target Type field is set to SUBNM.

Attributes Editor

Use the Attributes Editor to create and edit attributes for cubes, dimensions, elements, and replications.

Note that all elements include a Format attribute, which defines how element values display in the Cube Viewer. The default Format attribute value is Unstyled.

File Menu

Menu Item	Description
Close	Closes the Attributes Editor.

Edit Menu

Menu Item	Description
Undo cell	Undoes the last cell action. This option applies only to individual cells. You cannot undo actions applied to a range of cells.
Cut	Cuts the contents of selected cells to the Clipboard.
Copy	Copies the contents of selected cells to the Clipboard.
Paste	Pastes the contents of the Clipboard to selected cells.
Add new attribute	Opens the New Attribute dialog box, from which you can create a new attribute for the elements in the dimension.
Delete selected attribute	Deletes a selected attribute. You must delete attributes individually; you cannot delete multiple attributes simultaneously.
Clear	Clears the contents of selected cells.
Edit Element Format	Opens the Number Format dialog box, from which you can assign Format attribute values.

Format Options

The Format option is available only when you select cells at the intersection of the Format column and element rows. Click the **Format** button to display the Number Format dialog box.

Select an option from the **Category** list box to specify a display format for the selected cells.

The following number formats are available:

Format Category	**Description**
General	This format displays numbers without commas separating digits to the left of the decimal point. Negative values are prefixed with a minus sign (-). Use the Precision option to specify the number of digits that follow the decimal point. Note that Rules-derived values return integers only when set to General format.
Fixed	This format displays numbers without commas separating digits to the left of the decimal point. Negative values are prefixed with a minus sign (-); users have the option to use parentheses for negatives if preferred. Use the Precision option to specify the number of digits that follow the decimal point.
Currency	This format displays numbers with the currency symbol specified in your Windows RegionalSettingsProperties, and uses commas to separate every third digit to the left of the decimal point. Negative values are prefixed with a minus sign (-). Use the Precision option to specify the number of digits that follow the decimal point.
Date	Displays a list of predefined date formats.
Time	Displays a list of predefined time formats.
Percentage	This format multiplies numbers by 100 and displays a following percent sign (%). Digits to the left of the decimal point do not use commas, and negative values are prefixed with a minus sign (-). Use the Precision option to specify the number of digits that follow the decimal point.
Scientific	This format displays numbers in scientific notation. Negative values are prefixed with a minus sign (-). Use the Precision option to specify the number of digits that follow the decimal point.
Custom	You can define a custom format expression as needed.

Format Category	Description
Precision	This option determines the number of decimal places to display for a selected format. If a value has more decimal places than the specified precision, it is rounded off for display purposes only; the entire value is stored in the TM1 database.

Audit Log Window

Use the Audit Log window to query and view records contained in the TM1 audit log.

The Audit Log window contains two main panels; the Query panel and the Results panel. Use these panels to search the audit log and view the records retrieved by your search.

Query Panel

Use the Query panel to build queries that search the TM1 audit log.

The Query panel toolbar contains a **Run Query** icon ▶ to query the audit log after you set the query options.

The query options are organized into the following groups:

- Date and Time
- Event Owner
- Event Type.

Date and Time Options

The Date and Time options include set the time period that you want to query.

Option	Description
Time Period	Contains a list of predefined time periods for the query. Select a predefined time period or select **Custom Time Period** to enable the Start and End time options.
Start Time	The start date/time for the query. This option is enabled only when you select **Custom Time Period** for the Time Period option. TM1 queries against all records written to the audit log on or after this date/time. Click … to open the calendar tool where you can select a date and time.

Option	Description
End Time	The end date/time for the query. This option is enabled only when you select **Custom Time Period** for the Time Period option. TM1 queries against all audit records up to the end time you specify. Click to open the calendar tool where you can select a date and time. The default end time is the current date and time.

Event Owner Options

The Event Owner options answer the question "Who caused this event". The owner of the event can be an actual TM1 user or a scheduled chore.

The Event Owner options include the following parameters:

Option	Description
All	Sets the query to search for audit events caused by any TM1 user or scheduled chore.
Client	Sets the query to search for audit events caused only by TM1 users. To search for events caused by a specific TM1 user, click the **Select Client** button . You can select a single client or multiple clients. The default is all clients.
Scheduled Chore	Sets the query to search for audit events caused only by scheduled chores. To search for events caused by a specific scheduled chore, click the **Select Scheduled Chore** button . You can select a single scheduled chore or multiple scheduled chores. The default is all scheduled chore.

Event Type Options

The Event Type options let you the select the type of object or event for which you want to search. For example, you can use these search options to "find unsuccessful login attempts" or "find events where a dimension was deleted".

Option	Description
All	Sets the query to search for both types of audit events; system-wide and object related events.
System-wide	Sets the query to search for only system-wide audit events. To search for a specific system-wide event, select the event from the list. The default setting searches for *all* system-wide events.
Object	Sets the query to search for only object type audit events. To search for a specific object event, use the options as follows: • **Object Type** - Limits the query to only a specific type of TM1 object. For example, events related only to dimensions. • **Object Name** - Allows you to select a specific object name. Click … to display a dialog box where you can select objects by name. **Note:** When you set the Object Type option to **Element**, the Object Name Selection button becomes disabled because the element list could be too large to display. To search for events related to a specific element, you must manually enter an element name using the following format: DimensionName:ElementName. For example: region:italy • **Event Type** - Limits the query to only a specific type of object event. The default setting searches for *all* object type events.

Results Panel

Use the Results panel to view and navigate the records retrieved by your search.

Results Panel Toolbar

The Results toolbar has the following buttons:

Action	Button	Description
Copy	📋	Copies the value in the currently selected cell to the Windows clipboard.

Action	Button	Description
Find	🔍	Opens the Find dialog box where you can search for text in the event records.
Export	⬇	Opens the Save As dialog box where you can save the event records to a file in one of the following formats: • XML • Comma delimited • Tab delimited

Results Grid

The Results panel includes a grid that displays the audit log records retrieved by the query. The retrieved records are organized into the following columns:

Column	Description
Date	Date and time of the event.
User	TM1 client (user) or scheduled chore that was responsible for causing the event.
Event Type/ Description	Brief description of the event.
Object Type	Type of TM1 object associated with the event.
Object Name	Name of the TM1 object associated with the event.
Details	Displays an icon to indicate that detailed information exists for the specific event. If an event has details, you can view the details by clicking on the **Details** icon ℹ for that record.

You can sort the records in the grid in ascending or descending order for any column by clicking on the column title.

Audit Log Details Window

The Audit Log Details window displays the sub-events for an audit log event that was displayed in the query results of the main Audit Log window.

Details Toolbar

The Details toolbar has the following buttons:

Button	Description
Copy	Copies the value in the currently selected cell to the Windows clipboard.
Find	Opens the Find dialog box where you can search for text in the event records.
Export	Opens the Save As dialog box where you can save the event records to a file in one of the following formats: • XML • comma separated • tab separated

Details Grid

The Details grid displays the sub-event detail records that belong to the parent event.

The detail records are organized into the following columns:

Column	Description
Date	Date and time of the event.
User	TM1 client (user) or scheduled chore that was responsible for causing the event.
Event Type/ Description	Brief description of the event.
Object Type	Type of TM1 object associated with the event.
Object Name	Name of the TM1 object associated with the event.

You can sort the records in the grid in ascending or descending order for any column by clicking on the column title.

Chore Setup Wizard

Use the Chore Setup Wizard to schedule a replication or process for synchronization or execution at a regular interval.

The Wizard consists of two screens:

- **Screen 1** - Select the replications and processes to be included in the chore.
- **Screen 2** - Specify the start time for the initial execution of the chore and the subsequent interval at which the chore should execute.

Screen 1 (Step 1)

Field	Description
Available list	Lists all replications and processes available for scheduling as chores.
Selected list	Lists the replications or processes selected for inclusion in the current chore.
Add ›	Click this button to move selected replications or processes from the Available list to the Selected list
Add All »	Click this button to move all replications or processes from the Available list to the Selected list.
Remove ‹	Click this button to move selected replications or processes from the Selected list to the Available list.
Remove All «	Click this button to move all replications or processes from the Selected list to the Available list.
Specify Values for Parameters	Click to open the Parameter Values dialog box, from which you can specify values for any parameters associated with the selected process.

Screen 2 (Step 2)

Field	Description
Chore Start Date and Time	Select a start date on the calendar and specify a start time in the Time field.
Chore Execution Frequency	Fill the appropriate fields to establish the interval at which the chore should be executed.
Chore Schedule is Active	Fill this box to activate the chore for execution at the specified start time and interval. Clear this box to activate the chore at a later time.

Clients/Groups Window

The Clients/Groups window lets you create and modify clients and user groups on a server.

Clients/Groups grid

The Clients/Groups grid displays client names as row headings and user groups as column headings. An 'X' at the intersection of a client name and user group indicates the group to which the user belongs. Users can belong to multiple groups.

The grid also includes several columns that display properties for clients on the server.

- The cell at the intersection of a client name and the Password column contains the password for the client.
- The cell at the intersection of a client name and the Expiration Days column contains the number of days for which the password is valid for the client. After this number of days elapses, the client can no longer log into the server with the assigned password. A client whose password is soon to expire begins receiving notification of the expiration five days before the expiration date.
- The cell at the intersection of the client name and the Status column indicates whether the client is active on the server.
- The cell at the intersection of the client name and the Max Connections column indicates the maximum number of connections that can be established to the server with the associated client name and password.

Security Menu

Menu Item	Description
Close	Closes the Clients/Groups dialog box.

Clients Menu

Menu Item	Description
Add New Client	Opens the Creating New Client dialog box, from which you can create a new client on the server.
Delete Client	Deletes the currently selected client from the server.
Disconnect Client	Disconnects the currently selected client from the server.
Set Password	Sets the password for the currently selected client.
Clear Password	Clears the password for the currently selected client.

Groups Menu

Menu Item	Description
Add New Group	Opens the Creating New Group dialog box, from which you can create a new user group on the server.
Delete Group	Deletes the currently selected user group from the server.

Clients/Groups Grid

You can enter data for clients directly in the Clients/Groups grid.

The grid includes several columns, as described in the following table.

Column	Description
Username	Displays the usernames of all clients on the server.
Password	Identifies whether a password is defined for a given client. You can click in a cell at the intersection of the Password column and a client row, then type a password to assign a password to the client. After entering a password, TM1 prompts you to re-enter the password for confirmation.
Expiration Days	Indicates the number of days that a given client's password is valid. To assign expiration for a client's password, click in the cell at the intersection of the Expiration Days column and the client row, then type an expiration value.
Max Connections	Identifies the maximum number of connections that can be made to the server by a given client. To assign a maximum number of connections for a client, click in the cell at the intersection of the Max Connections column and the client row, then type the maximum number of connections for the client.
Status	Indicates the current connection status of a given client.
User Groups	There is one column for every user group on the server. To assign a client to a user group, fill the check box at the intersection of the user group column and the client name. Clients can belong to multiple user groups.

Clients Messaging Center Dialog Box

The Clients Messaging Center dialog box lets you manage client connections to a server. You can also use this dialog box to remotely shut down a server. You must be a member of the ADMIN group for a server to access this dialog box.

Select a server in the left pane of the Server Explorer, then choose **Server**, **Server Manager** to open the Clients Messaging Center dialog box.

Field	Description
Shutdown Server	Select this option to shut down the server, then specify a Minutes interval.
Disconnect Clients	Select this option to disconnect clients from the server, then specify a Minutes interval. You must click Select Clients to create or select a subset of clients to be disconnected.
Broadcast Message to Selected Clients	Select this option to broadcast a text message to clients connected to the server. Enter the message in the text box then click Select Clients to create or select a subset of clients to receive the message.

Create a Dimension Dialog Box

Enter a name for the dimension you want to create in the field at the top of the dialog box then click **OK**.

To create a dimension on your local server, enter only the dimension name.

To create a dimension on a remote server, prefix the dimension name with the server name and a colon. For example, enter **Sales:Product** to create the Product dimension on the Sales server.

Create Server Replication Object Dialog Box

Use the Create Server Replication Object dialog box to establish a new replication connection, or to modify an existing connection.

Field	Description
To Server	Select a source server from the list. The list includes the names of all servers currently available on your network.
As User	Enter your user name on the selected source server.
With Password	Enter your password for the selected source server.

Field	Description
With Namespace	If the object uses CAM Passport security, enter the IBM Cognos Namespace ID. Do not enter the descriptive name here.
Use Integrated Login	Check this box to use Integrated Login authentication instead of standard TM1 security.

Creating Cube Dialog Box

Use the following options on the Creating Cube dialog box to create a new cube from previously-defined dimensions.

Field	Description
Cube Name	Type the name for the cube you are creating in this field.
Available Dimensions	A list of all dimensions available on the server on which you are creating the cube.
Dimensions in New Cube	The list of dimensions in the cube you are creating.
Add	Click this button to move selected dimensions from the Available Dimensions list to the Dimensions in New Cube list
Remove	Click this button to move selected dimensions from the Dimensions in New Cube list to the Available Dimensions list.
Move up	Click this button to move selected dimensions up through the Dimensions in New Cube list. Each click of the button moves the selected dimensions up one position.
Move down	Click this button to move selected dimensions down through the Dimensions in New Cube list. Each click of the button moves the selected dimensions down one position.
Cancel	Click to cancel the cube creation and exit the Creating Cube dialog box.
Reset	Click to reset the Available Dimensions list and clear the Dimensions in New Cube list.
Refresh	Click to refresh the Available Dimensions list. This option polls the server for any new dimensions, and adds any new dimensions to the Available Dimensions list.
Properties	Click this button to assign cube properties. You can assign properties that define a measures dimension, a time dimension, and load-on-demand status for the cube.

Field	Description
OK	Click to accept the configuration of the dialog box and create the cube.

Cube Optimizer Dialog Box

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 server to reconfigure the order of dimensions in a cube. During the re-ordering process, the temporary RAM on the 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**.

Cube Properties Dialog Box

Use the Cube Properties dialog box to set properties for individual cubes.

Field	Description
Measures Dimension	Select a measures dimension from the list.
Time Dimension	Select a time dimension from the list.
Load on Demand	Fill the box to load the cube into server memory only when a client requests cube data. Clear this box to load the cube automatically when the server starts.

Cube Viewer

Title dimensions

Title dimensions appear directly beneath the Toolbar at the top of the Cube Viewer window. Each dimension displays in a list box.

Row dimensions

Row dimensions appear at the top of the row axis of the Cube Viewer. The current dimension elements appear as row headings in the Cube Viewer.

Column dimensions

Column dimensions appear at the left of the column axis of the Cube Viewer. The current dimension elements appear as column headings in the Cube Viewer.

File Menu

The following options are available on the File Menu in the Cube Viewer.

Option	Description
Open	Opens the TM1 Open View dialog box, from which you can open other views associated with the current cube.
Reload	Reloads the current view definition.
Calculate	Calculates the current view.
Save	Saves the current view configuration.
Save as	Saves the current view configuration under a new name.
Delete Views	Opens the Delete Named Views dialog box, from which you can delete saved views.

Option	Description
Slice	Exports the current view into an Excel worksheet. The Excel worksheet is populated with formulae that retrieve values from and write values to the server from which the view originates.
Active Form	Launches the Insert Active Form option to let you add an Active Form connection to data in the current cell of the worksheet.
Snapshot	Exports the current view to an Excel worksheet as simple values. The worksheet does not maintain a connection to the server from which the view originates.
Close	Closes the Cube Viewer window.

Edit Menu

The following options are available on the Edit Menu in the Cube Viewer.

Option	Description
TransAction	Undoes the last cell action. Save or Close ends the collection of actions that can be undone or redone. Redo restores the last cell action.
Cut	Cuts the contents of selected cells to the Clipboard.
Copy	Copies the contents of selected cells, as currently formatted, to the Clipboard.
Copy Unformatted Value	Copies the unformatted contents of selected cells to the Clipboard.
Paste	Pastes the contents of the Clipboard to selected cells.
Delete	Deletes the selected cell values.
Edit Cube Attributes	Opens the Attributes Editor window, from which you can assign and edit attributes for all cubes on the current server.

View Menu

The following options are available on the View Menu in the Cube Viewer.

Option	Description
Toolbar	Hides or displays the Toolbar at the top of the Cube Viewer. A check mark indicates that the Toolbar is displayed.
Status Bar	Hides or displays the Status Bar at the bottom of the Cube Viewer. A check mark indicates that the Status Bar is displayed.

Option	Description
Right to Left	This toggle changes the position of column dimensions in the Cube Viewer. A right pointing arrow indicates that the columns layout right to left. A left pointing arrow means columns are laid out left to right.

Options Menu

The following options are available on the Options Menu in the Cube Viewer

Option	Description
Suppress Zeros	This option suppresses or displays all rows and columns containing only zero values in the cube view. A check mark indicates that rows and columns containing only zeros are suppressed in the current view.
Suppress Zeros on Rows	This option suppresses or displays all rows containing only zero values in the cube view. A check mark indicates that rows containing only zeros are suppressed in the current view.
Suppress Zeros on Columns	This option suppresses or displays all columns containing only zero values in the cube view. A check mark indicates that columns containing only zeros are suppressed in the current view.
Automatic Recalculate	This option enables or disables automatic recalculation upon view reconfiguration. A check mark indicates that the view is automatically recalculated whenever the view configuration changes.
Format	Opens the Number Format dialog box, from which you can define the number format for values in the current view. Note that the format you select applies only to those values for which there is no Format attribute specified.
Column Width	Opens the Column Width dialog box, which lets you set a minimum and maximum width for columns in the Cube Viewer.
Slice to New Workbook	This option determines how slices are created. A check mark indicates that slices are inserted in a new workbook when you choose File, Slice. If this option is not turned on, slices are inserted in a new sheet of the current workbook.

Delete Named Subsets Dialog Box

This dialog box displays the subsets associated with the current dimension. To delete a subset, select the subset and click OK.

To select multiple adjacent subsets, click and drag across the subsets. To select multiple non-adjacent subsets, CRTL-click each subset.

Delete Named Views Dialog Box

This dialog box displays the views associated with the current cube. To delete a view, select the view and click OK.

To select multiple adjacent views, click and drag across the views. To select multiple non-adjacent views, CRTL-click each view.

Dimension Editor

Elements Pane

Displays elements of the dimension you are currently viewing.

Properties Pane

When you select a consolidated element in the Elements pane, the Properties pane displays the properties of the immediate children of the consolidated element.

When you select a leaf element, the Properties pane displays the properties of the leaf element.

Note: When viewing an exceptionally large dimension set in the Dimension Editor with the Properties pane on, you might experience performance issues. This can happen when you select a consolidation in the Elements pane and TM1 has to display the entire list of related elements and properties in the Properties pane.

If you are working with large dimension sets, you may want to turn off the Properties pane. To turn off the Properties pane, click the **Properties Window** option in the View Menu to remove the check mark next to the option.

Dimension Menu

Menu Item	Description
Save	Saves the current dimension structure.
Save as	Saves the current dimension structure under a new name.
Close	Closes the Dimension Editor.

Edit Menu

Menu Item	Description
Cut	Cuts selected elements to the Clipboard.

Menu Item	Description
Copy	Copies selected elements to the Clipboard.
Paste	Pastes the contents of the Clipboard as a new element. • When no elements are selected in the Dimension Editor, this option inserts a new element above the first displayed element in the Elements pane. • When an element is selected in the Elements pane, this option displays a sub-menu with the options Paste Above, Paste as Child, and Paste Below.
Paste Above	Pastes the contents of the Clipboard above a selected element.
Paste Below	Pastes the contents of the Clipboard below a selected element.
Paste as Child	Pastes the contents of the Clipboard as a child of a selected element.
Insert Child	Opens the Dimension Element Insert dialog box, from which you can insert a child or children of a selected element.
Insert Element	Opens the Dimension Element Insert dialog box, from which you can insert leaf (simple) elements into the dimension.
Select All	Selects all the elements in the Elements pane.
Filter by, Level	Opens the Filter by Level dialog box, from which you can select elements by hierarchy level. This option affects only the display of elements; it does not affect the dimension structure. When you use this option the Elements pane displays only the elements of the level you specify.
Filter by, Attribute	Opens the Filter by Attribute dialog box, from which you can select elements by attribute value. This option affects only the display of elements; it does not affect the dimension structure. When you use this option the Elements pane displays only those elements with the attribute value you specify.

Menu Item	Description
Filter by, Wildcard	Lets you select elements that match a user-defined search expression. This option affects only the display of elements; it does not affect the dimension structure. When you use this option the Elements pane displays only those elements matching the search expression you specify.
Select Alias	Opens the TM1 Aliases dialog box, from which you can select an alias to use for display in the Dimension Editor.
Sort, Ascending	Sorts all elements in the Elements pane in alphabetically ascending order. This option affects only the display of elements; it does not affect the dimension structure.
Sort, Descending	Sorts all elements in the Elements pane in alphabetically descending order. This option affects only the display of elements; it does not affect the dimension structure.
Sort, Hierarchy	Sorts all elements in the Elements pane in hierarchical order, so you can see the parent/child relationship of elements. This option affects only the display of elements; it does not affect the dimension structure.
Sort, Index Ascending	Sorts all elements in the Elements pane in ascending order according to element index value. This option affects only the display of elements; it does not affect the dimension structure.
Sort, Index Descending	Sorts all elements in the Elements pane in descending order according to element index value. This option affects only the display of elements; it does not affect the dimension structure.
Keep	Alters the Elements pane so that only currently selected elements are displayed. This option affects only the display of elements; it does not affect the dimension structure.

Menu Item	Description
Hide	Alters the Elements pane so that currently selected elements are hidden. This option affects only the display of elements; it does not affect the dimension structure.
Delete Element	Deletes all instances of a selected element from the dimension.
Delete from Consolidation	Deletes the instance of a selected element from the current consolidation.
Edit Element Formats	Opens the Edit Element Formats worksheet, from which you can define element display styles. These display styles are applied in dynamic slices and in TM1 Web websheets.
Expand Element	Displays all children of a selected element.
Collapse Element	Hides all children of a selected element.
Properties	Opens the Dimension Element Properties dialog box, from which you can assign element type and weight for a selected element.

View Menu

Menu Item	Description
Toolbars	Hides or displays the various toolbars at the top of the Dimension Editor window. A check mark indicates that a toolbar is displayed.
Status Bar	Hides or displays the Status Bar at the bottom of the Dimension Editor window. A check mark indicates that the Status Bar is displayed.
Properties Window	Hides or displays the Properties pane. A check mark indicates that the Properties pane is displayed.
Refresh	Updates the display of the Elements pane.

Dimension Element Insert Dialog Box

Use this dialog box to add simple, string, or consolidated elements to a dimension. The dialog contains the following options.

Option	Description
Dimension Name	The name of the dimension to which you are adding elements. This is not an editable option.
Parent Name	The name of the parent element to which you are adding elements. This is not an editable option. If an element was selected in the dimension editor when you opened the Dimension Element Insert dialog box, that element displays as the Parent Name. If no element was selected, the Parent Name is Root.
Insert Element Name	Enter a name for the new element in this box.
Element Type	Make a selection appropriate to the element you want to insert.
Element Weight	If the element type is Simple and the Parent Name is anything other than Root, enter a weight in this box. The weight is a multiplication factor applied to an element during consolidation. A weight associated with an element of a consolidation does not alter the value of the element elsewhere in the dimension.
Add	Click Add each time you specify a new element, type, and weight.
OK	Click this button when you are done adding elements to commit the new elements to the dimension.

Dimension Element Ordering Dialog Box

Use this dialog box to set the order of elements in a dimension.

The order of elements within a dimension determines the index value for each element in the 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. The order of elements in a dimension is important because many TM1 functions (worksheet, rules, and TurboIntegrator) reference 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.

Use the following steps to set the order of elements.

Procedure

1. Select a sort type.

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.

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

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

4. If applicable, select a **Sort Direction**.
5. Click **OK**.

   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 3.

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 3.

Dimension Element Properties Dialog Box

Displays the name, type, and weight of the current element.

Properties Pane

Options	Description
Element Type	To change the type of the current element, select a new type from the list. There are three possible element types: simple, consolidated, and string.
Element Weight	To change the weight of the current element, double-click in the Element Weight field and enter a new weight value.

Drill

The Drill menu lists the options used to create and manage a drill process and drill assignment. Drill processes and assignments are used to create links between cube cells with related detailed data.

Options	Description
Create/Edit/Delete Drill Assignment Rules	Choose these options to create, edit or delete drill assignments. The Create option opens the rules editor so you can design the rule.
Create/Edit Drill Process	A drill process is a TurboIntegrator process that defines the detailed data, which opens in a new window. These options edit an existing drill assignment rule or allow you to create a new one. The Create options display the parameters and values to use and the details for the data source. 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.

Edit Formula Dialog Box

The Edit Formula dialog box steps you through the creation of DBR, DBRW, and DBS functions. You can also use the Edit Formula dialog box to edit any TM1 function in a worksheet.

To display the Edit Formula dialog box, click a cell in a worksheet and choose **TM1 , Edit Formula**. If the cell contains a TM1 function, the function displays in the entry field of the dialog box.

Field	Description
DB Ref	Click this button to insert a DBR function in the current cell. TM1 steps you through several dialog boxes that help you create the function.
DBRW	Click this button to insert a DBRW function in the current cell. TM1 steps you through several dialog boxes that help you create the function.
DB Send	Click this button to insert a DBS function in the current cell. TM1 steps you through several dialog boxes that help you create the function.

Field	Description
Cell Ref	Click this button to insert a cell reference into a function. TM1 prompts you to select the cell to which you want to refer, and prompts for a reference type.
Names	Click this button to insert a cube, dimension, or element name into a function

The Formula Editor can be used to create functions that reference cubes of up to 29 dimensions.

Edit Reference to Cube Dialog Box

This dialog box lets you set the element references used in TM1 worksheet functions such as DBRW and DBSW.

The dialog box contains buttons and fields corresponding to each dimension in the cube that the TM1 worksheet function references. For example, the following image shows the Edit Reference to Cube dialog box for a DBRW function that references the SalesCube cube in the TM1 sample database. The dialog box includes buttons for all the dimensions in the SalesCube cube.

When you insert a TM1 function into a worksheet, TM1 attempts to determine if any relevant element references exist in the worksheet. If so, those references are automatically inserted into the appropriate fields on the Edit Reference to Cube dialog box. If relevant element references cannot be determined, TM1 inserts "Undef" in the fields.

You can set references in this dialog box by either:

- clicking a dimension button and selecting an element. In this case, the reference is inserted as a string into the appropriate field.
- entering a cell reference directly in a field. You can use row-relative, column-relative, or absolute cell references.

If the cube for which you are creating a reference contains more than 16 dimensions, click **Previous** to page backward to the previous 16 dimensions, or click **Next** to page forward to the next 16 dimensions.

Filter Elements by Attribute Dialog Box

Use this dialog box to select only those subset elements that have a specified attribute value.

Select the desired attribute from the Select an Attribute list.

Select a corresponding value from the Select a Value list.

Filter Elements by Level Dialog Box

The list box displays the hierarchy levels available in the current subset. To view only elements of a given level, select the level and click OK.

To select multiple adjacent levels, click and drag across the levels. To select multiple non-adjacent levels, CRTL-click each level.

Filter Subset Dialog Box

The Filter Subset dialog box lets you create a dynamic subset based on values in a specified cube. For example you can create a subset of the Region dimension that returns the 10 elements with the largest values for actual yearly sales of the 1.8L Sedan in the Sales cube.

The dialog box contains the following options.

Option	Description
CubeName	The cube for which you want to filter values.

Option	Description
Filter	The type of filter you want to apply to the current view. **TopCount** Filters the subset to return only the largest n elements, where n is a number specified in the Value option. **BottomCount** Filters the subset to return only the smallest n elements, where n is a number specified in the Value option. **TopSum** Filters the subset to return only the largest elements whose sum is greater than or equal to *n*, where *n* is a number specified in the Value option. **BottomSum** Filters the subset to return only the smallest elements whose sum is greater than or equal to *n*, where *n* is a number specified in the Value option. **TopPercent** Filters the subset to return only the largest elements whose sum is greater than or equal to *n*, where *n* is a percentage of the dimension total specified in the Value option. **BottomPercent** Filters the subset to return only the smallest elements whose sum is greater than or equal to *n*, where *n* is a percentage of the dimension total specified in the Value option. **None** Not applicable to filtering subsets.
Value	A value for the Filter type.
Select Column Member	The column element(s) against which the filter or sort is applied. Click the dimension buttons to select a single element for each column dimension.

Option	Description
Sort	The sort order you want to apply to the selected column element(s). **Ascending** Sorts values for the specified column element(s) from lowest to highest. **Descending** Sorts values for the specified column element(s) from highest to lowest. **None** No sort order.
Select Column Members	You must select a single element from each remaining cube dimension. For example, if you are filtering the Region dimension in the sample database against values in the Sales cube, you must specify a single element each of the Model, Month, ActVsBud, and Account1 dimensions. For each dimension, click the appropriate button and select a single element. If the cube contains more than 16 dimensions, click ▲ to page backward to the previous 16 dimensions, or click ▼ to page forward to the next 16 dimensions.

Filter View Dialog Box

The Filter View dialog box lets you filter and sort columns in the Cube Viewer or In-Spreadsheet Browser.

The dialog contains the following options.

Option	Filter/Description
CubeName	The cube for which you want to filter or sort values. This option is always set to the cube associated with the current view. It cannot be edited.
Filter	The type of filter you want to apply to the current view.
	TopCount Filters the view to display only the largest n elements, where n is a number specified in the Value option.

Option	Filter/Description
	BottomCount Filters the view to display only the smallest n elements, where n is a number specified in the Value option.
	TopSum Filters the view to display only the largest elements whose sum is greater than or equal to *n*, where *n* is a number specified in the Value option.
	BottomSum Filters the view to display only the smallest elements whose sum is greater than or equal to *n*, where *n* is a number specified in the Value option.
	TopPercent Filters the view to display only the largest elements whose sum is greater than or equal to *n*, where *n* is a percentage of the dimension total specified in the Value option.
	BottomPercent Filters the view to display only the smallest elements whose sum is greater than or equal to *n*, where *n* is a percentage of the dimension total specified in the Value option.
	None No filter. Select this option if you want to sort values without filtering.
Value	A value for the Filter type.
Select Column Member	The column element(s) against which the filter or sort is applied. Click the dimension buttons to select a single element for each column dimension.
Sort	The sort order you want to apply to the selected column element(s).
	Ascending Sorts values for the specified column element(s) from lowest to highest.
	Descending Sorts values for the specified column element(s) from highest to lowest.
	None No sort order.

Get View Dialog Box (In-Spreadsheet Browser)

The Get View dialog box lets you open a view on your local server or on any servers available on your network.

Field	Description
Server	The Server list displays all servers available on your network. Select the server on which the view you want to open resides. If you are not logged on to the server containing the view you want to open, click Connect to open the Connect Server dialog box and log on to the server. Click Start Local Server to start your local server.
Cube	The Cube list displays all cubes available on the selected server. Select the cube associated with the view you want to open.
View	The View list displays all views available on the selected cube. Select the view you want to open.

In-Spreadsheet Browser Menu

The In-Spreadsheet Browser Menu is available from a right-click on the TM1 View Control. The menu lets you open, update, format, slice and save a view. It also includes several options that control the behavior of the In-Spreadsheet Browser.

Menu Item	Description
Update View	Updates the current view by sending any edited values to the TM1 database and retrieving current values from the database.
Get View	Opens the Get View dialog box, from which you can open a view on any available server.
Styles	Opens the View Styles dialog box, which lets you format a view.
Save	Opens the Save View dialog box, which lets you save a TM1 view.
Clear Display	Clears all data associated with a view, including title, row, and column labels.
Delete	Deletes the TM1 View Control. Note that all data associated with the view, including values and labels, remain in the spreadsheet.
Cut	Cuts the TM1 View Control to the Clipboard.
Copy	Copies the TM1 View Control to the Clipboard.
Slice	Slices the current view into a new Excel spreadsheet.
Suppress Zeroes	This toggle suppresses or displays zero values in the cube view. A check mark indicates that zeros are suppressed in the current view.

Menu Item	Description
Show Automatically	This toggle enables or disables automatic view update upon view reconfiguration. A check mark indicates that the view is automatically updated whenever the view configuration changes.
Update View on Recalc	This toggle enables or disables automatic view update upon spreadsheet recalculation (F9). A check mark indicates that the view is updated whenever the spreadsheet is recalculated.
Help	Open the In-Spreadsheet Browser help topic.

Message Log Window

The TM1 Message Log window displays status messages on the activity of the server. These messages are saved to the server message log and contain details on activity such as executed processes, chores, loaded cubes and dimensions, and synchronized replication.

For detailed information about the server message log, see the IBM Cognos TM1 *Operation Guide*.

Message Log pane

This pane displays status messages contained in the server message log.

Each row in the pane represents one unique message. If a message in the log shows an error condition for an executed process or replication, you can double-click the message to view the details of why the activity generated the error.

For details about the fields in the Message Log pane, see the IBM Cognos TM1 *Operation Guide*.

File Menu

Menu Item	Description
Exit	Closes the Message Log window.

Edit Menu

Menu Item	Description
Copy	Copies the selected text from the Message Log pane to the Clipboard.
Find	Opens the Find dialog box where you can search for text in the Message Log pane.

Help Menu

Menu Item	Description
Message Log Help	Opens the Message Log help topic.
Contents and Index	Opens the full TM1 Documentation Library.

New Attribute Dialog Box

Field	Description
New Attribute Name	Enter a name for the new attribute in this field.
Numeric	Select this option if the attribute values are numbers.
String	Select this option if the attribute values are character strings.
Alias	Select this option if the attribute values are alternative names for current element, dimension, cube, or server names.

Open Subset Dialog Box

Use the Open Subset Dialog Box to open an existing dimension subset.

To open the public default subset, select the Default box and click **Open**.

Open View Dialog Box

Use the Open View Dialog Box to open an existing cube view.

To open the public default view, select the Default box and click **Open**.

Print Report Wizard

Use the Print Report Wizard to generate "briefing book"-style reports from TM1 slices.

The Wizard consists of three screens.

- Screen 1 - Select the sheets to include in the report
- Screen 2 - Select the title dimensions to use in the report, set the order in which they appear in the report, and set workbook print options
- Screen 3 - Select a print destination for the report (printer, Excel file, or PDF file)

The Print Report Wizard also allows you to save your report settings.

All Screens

Button	Description
Load	Click this button to load an existing TM1 Print Job.
Save	Click this button to save the current report settings as a TM1 Print Job.
Save As	Click this button to save the current report settings as a TM1 Print Job under a new name.
Next	Click this button to advance to the next Wizard screen.
Cancel	Click this button to close the Wizard window without generating a report.

Screen 1 of 3

Item	Description
Include these sheets in the report list	Lists the available worksheets in the current Excel workbook that you can include in the report. To include a worksheet in the report, select the check box next to the sheet name.
Select All	Click this button to include all sheets in the report.
Clear All	Click this button to exclude all sheets from the report.

Screen 2 of 3

Item	Description
Available Title Dimensions list	Lists the available title dimensions that you can use in the report. For each dimension, this list displays the subset name (if applicable), number of elements in the dimension or subset, and cell address of the title dimension in the worksheet.
Selected Title Dimensions list	Lists the title dimensions to include in the report. The order of this list is used when TM1 generates the report.

Item	Description
Add ›	Click this button to move selected dimensions from the Available Title Dimensions list to the Selected Title Dimensions list.
Add All »	Click this button to move all dimensions from the Available Title Dimensions list to the Selected Title Dimensions list.
Remove ‹	Click this button to move selected dimensions from the Selected Title Dimensions list to the Available Title Dimensions list.
Remove All «	Click this button to move all dimensions from the Selected Title Dimensions list to the Available Title Dimensions list.
Move Up ▲	Click this button to move the selected dimension up in the Selected Title Dimensions list. The order in this list is used when TM1 generates the report.
Move Down ▼	Click this button to move the selected dimension down in the Selected Title Dimensions list. The order in this list is used when TM1 generates the report.
Subset Editor	Click this button to open the Subset Editor if you want to select a subset of elements from the currently selected dimension in the Selected Title Dimensions list.
Print Single Workbook	Select this option to create a report arranged into one complete group of worksheets. Each sheet in the report is printed only once, including sheets that do not contain TM1 slice data.
Print Multiple Workbooks	Select this option to create a report arranged into multiple groups based on dimension elements. This option creates a report with a larger number of sheets because a copy of each sheet is printed for each title element.
Total Excel Workbooks that will be generated	Displays the total number of Excel sheets that TM1 will generate for the current report.

Screen 3 of 3

Field	Description
Print to Printer	Select this option if you want to print the report to a printer.
Save As Excel Files	Select this option if you want to generate the report as an Excel file.
Save As PDF Files	Select this option if you want to generate the report as a PDF file.
Preview	This button becomes available when you select the Print to Printer option. Click this button to preview the report before printing.
Printer Name	This option becomes available when you select the Print to Printer option. Use this option to specify the printer to which TM1 prints the report.
Number of Copies	This option becomes available when you select the Print to Printer option. Use this option to specify the number of copies of the report to print.
Print To File	This option becomes available when you select the Print to Printer option. Select this option to save the report as a printer-ready file.
File Name	This option becomes available when you select both the Print to Printer and Print to File options. Enter a full path and file name to which you want to save the report. You must also specify a file type. For example, if you print to a file using a PostScript printer, you should append the .ps file type to the file name.
Browse	This button becomes available when you select the option to print or save the report to a file. Click this button to choose the directory in which you want to save the report.

Field	**Description**
Collate	This option becomes available when you select the Print to Printer option. Select this option to group pages together when printing multiple copies of the report.
Generate New Workbook for Each Title	This option becomes available when you choose to save the report as an Excel or PDF file. Select this option if you want to create a separate file for each title dimension in the report.
Directory Name	This option is available when saving a report as an Excel or PDF file and you select the Generate New Workbook for Each Title option. Enter a directory in which to save the report files. To choose a directory location, click the Browse button.
Create Snapshot	This option becomes available when you select the Save As Excel Files option. Select this option when you want to save the report as an Excel file that contains actual values and not TM1 functions that retrieve values.
Back	Click this button to step back to the previous Wizard screen.
Finish	Click this button to generate the report based on the options you have selected.

Process Options Dialog Box

Use the Process Options dialog box to control the behavior of the Action button *before* and *after* the process is run.

You can use one of the following methods to set the text for confirmation and status messages that display when the Action button is clicked:

- Enter text for a message directly into the text box.
- Use an Excel reference to dynamically retrieve the text for a message from the worksheet.

For example, to retrieve the text for a message from the contents of cell A1, enter =A1 into the text box for that message. To reference a named range, use the format: *=Named Range*.

For more information about using the Process Options dialog, see the IBM Cognos TM1 *Developer Guide*.

Field	Description
Automatically Recalculate Sheet	Select this option to have TM1 automatically recalculate the worksheet after the process has run.
Show Success Message	Select this option to display a message *after* the process has run successfully. Enter your message text into the box as described above.
Show Failure Message	Select this option to display a message if the process does not run successfully. Enter your message text into the box as described above.
Show Confirmation Dialog	Select this option to display a Yes/No confirmation message box *before* the process starts. The user can click either Yes, to run the process, or No, to cancel. Enter your message text into the box as described above.
OK	Click this button to save your settings and close the dialog box.
Cancel	Click this button to close the dialog box without saving your settings.

Replicate Cube Dialog Box

Use the Replicate Cube dialog box to replicate a cube from a source server to a target server.

Cube Information

Item	Description
Name	The name of the mirror cube on the target server. By default, TM1 names the mirror cube by concatenating the source server name with the source cube name. Do not change the default name if you are replicating rules in that cube.
Copy Data and Set to Synchronize	Select this option to copy data when the replication is established and to synchronize data when synchronization occurs between the source and target servers.

Item	Description
Copy Data but Do Not Set to Synchronize	Select this option to copy data when the replication is established but to disable later synchronization of data.
Replicate Views	Select this option to replicate all views associated with the source cube.

Rule Information

Item	Description
Copy Rule	Select this option to copy any rules from the source cube to the mirror cube.
Set Rule to Synchronize	Fill this box to synchronize rules when synchronization occurs between the source and target servers. Clear this box to disable synchronization of the rule.
Do Not Copy Rule	If you select this option, TM1 does not copy the rule from the source cube to the mirror cube.

Dimension Information

Item	Description
Dimension Information box	This box displays information about the dimensions in the mirror cube. If the source cube does not contain rules, TM1 renames the mirror dimensions by concatenating the source server names with the source dimension names. If the source cube contains rules, TM1 does not change the dimension names in the mirror cube. The Dimension Information box also displays the name of the source dimension, source server, and replication status for each dimension in the cube.
Select Local Dimension	To use a local dimension in the place of a source dimension, click the source dimension in the Dimension Information box and click Select local dimension. Select the local dimension you want to use and click OK.

Item	Description
Reset Current Selection to Default	If you change any Dimension Information options for a dimension in a replicated cube, you can restore all options to default values by selecting the dimension in the Dimension Information box and clicking this button.
Overwrite Dimension	This option becomes available when you select a local dimension. Select this option to overwrite the local dimension with the definition of the source dimension.
Set Dimension to Synchronize	Fill this box to synchronize changes to between the source and mirror dimension when synchronization occurs between the source and target servers. Clear this box to disable synchronization of the dimension.
Don't overwrite dimension	This option becomes available when you select a local dimension. Select this option to use the local dimension as-is.
Replicate Subsets	Select this option to replicate all subsets associated with the source dimension.

Rules Editor

The Rules Editor has a full set of menus for creating, editing, and managing TM1 rules. Keyboard shortcuts are provided for the more commonly used menu options.

File Menu

The following table describes the options in the File Menu.

Name	Description
Import	Opens a file browse dialog so you can select a text file to import. Imported text will overwrite the current rule if one exists.
Save	Saves the current rule to the server.
Save As	Saves the current rule to an external TM1 rule .rux file.
Check Syntax	Checks the current rule for syntax errors.

Name	Description
Print	Opens the Print dialog box so you can print the current rule.
Print Preview	Opens the Print Preview window where you can view a sample printed version of the rule before sending it to a printer.
Exit	Closes the Rules Editor.

Edit Menu

The following table describes the options in the Edit Menu.

Name	Description
Undo	Undoes the last edit. Multiple levels of undo are supported.
Redo	Reverses the last undo command.
Cut	Removes the selected text and places it in the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes the contents of the clipboard into the Rules Editor.
Select All	Selects the entire contents of the Rules Editor.
Find	Opens the Find dialog box so you can search for text in the rule.
Find / Replace...	Opens the Find/Replace dialog box to search for and replace text.
Find Next	Locates the next occurrence of the text for which you are searching.
Toggle Bookmark	Turns a bookmark on or off for the current line of code.
Next Bookmark	Moves the cursor to the next available bookmark.
Previous Bookmark	Moves the cursor to the previous available bookmark.
Clear All Bookmarks	Removes all bookmarks.

Name	Description
Comment Selection	Adds a comment symbol # in front of all lines in the currently selected text to exclude the lines from the compiled rule. **Note:** Comment length is limited to 255 bytes. For Western character sets, such as English, a single character is represented by a single byte, allowing you to enter comments with 255 characters. However, large character sets, such as Chinese, Japanese, and Korean, use multiple bytes to represent one character. In this case, the 255 byte limit may be exceeded sooner and not actually allow the entry of 255 characters.
Uncomment Selection	Removes the comment symbol # from in front of all lines in the currently selected text to include the lines in the rule.
Indent	Indents the currently selected lines.
Unindent	Removes the indent from the currently selected lines.
Goto Line...	Displays the Go To Line dialog box so you can enter and jump to a specific line number in the Rules Editor.

View Menu

The following table describes the options in the View Menu.

Note: Any changes you make to the settings on the View Menu are saved when you exit the Rules Editor and are automatically re-applied the next time you open the Rules Editor.

Name	Description
Word Wrap	Turns on/off the word wrap feature so lines of text either extend to the right or wrap to display within the Edit pane.
Line Numbers	Turns on/off line numbers.
Function Tooltips	Turns on/off the display of function tooltips.
Auto-Complete	Turns on/off the auto-complete feature when typing in the Edit pane.
Toolbar	Turns on/off the display of the main toolbar.
Status Bar	Turns on/off the display of the status bar at the bottom of the Rules Editor.
Control Objects	Turns on/off the display of TM1 control objects when selecting cubes.

Name	Description
Expand All Regions	Expands all of the user-defined regions in the current rule to show all lines.
Collapse All Regions	Collapses all of the user-defined regions in the current rule to hide all lines that are included in a region.

Insert Menu

The following table describes the options in the Insert Menu.

Name	Description
Function	Displays the Insert a Function dialog box to enter a new function into the current rule.
Cube Reference	Displays the Insert Cube Reference dialog so you can insert a DB function.

Tools Menu

The following table describes the options in the Tools Menu.

Name	Description
Preferences...	Displays the Preferences dialog where you can set the font attributes such as font type, size, and color to be used in the Edit pane.
Options...	Displays the Control Options dialog where you can adjust the global settings for the Rules Editor.

Save Subset Dialog Box

Field	Description
Select or Enter Subset Name	Enter a name for the saved subset, or select a name from the list.
Private	Toggle this option on to save the subset as a private object. Toggle this option off to save the subset as a public object.
Default	Toggle this option on to save the subset as a default subset.

Field	Description
Save Expression	If the subset is dynamic, toggle this option on to save the MDX expression with the subset. If the subset is dynamic and you do not toggle this option on, the MDX expression is not saved and the resulting subset is static, containing the elements present when saved.

Save View Dialog Box

Field	Description
Select or Enter Named View	Enter a name for the saved view, or select a name from the list.
Private	Toggle this option on to save the view as a private object. Toggle this option off to save the view as a public object.
Default	Toggle this option on to save the view as a default view.

Save View Dialog Box (In-Spreadsheet Browser)

Field	Description
View Name	Enter a name for the view in this field.
Private	Toggle this option on to save the view as a private object. Toggle this option off to save the view as a public object.
Default	Toggle this option on to save the view as a default view.

Security Assignments Dialog Box

The Security Assignments dialog box lets you assign access privileges for cubes, dimensions, individual elements, processes, and chores. Access privileges are assigned by user group.

Assignments Grid

The Assignments grid displays object names as row headings and user groups as column headings. Access privileges appear as cell values at the intersection of a given object and user group.

When you access the Security Assignment dialog box from a Cubes group, the grid includes a Logging column. This column includes a check box for each cube. To enable logging for a cube, turn on the check box at the intersection of the cube name and the Logging column. To disable logging, turn off the check box. The default is on.

Access Privileges

Click one of the following options to assign an access privileges to a selected cell in the Assignments grid:

None Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned the None privilege for an object.

Object	Description
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 view 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 any cubes that contain the dimension.
Process	Members of the group cannot see the process in the Server Explorer. **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.
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.

Read Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned Read privilege for an object

Object	Description
Cube	Members of the group can view data in the cube, but cannot edit the data.
Element	Members of the group can view data identified by the element, but cannot edit the data.

Object	Description
Dimension	Members of the group can view the elements in a dimension, but cannot edit the dimension structure.
Process	Members of the group can see the process in the Server Explorer and can 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 or change the activation status.
Application	Members of the group can see the application and use any references within the application to which you have at least Read privilege. You can create private references in the application, as well as private sub-applications
Reference	Members of the group can open and use the reference, but cannot update the reference in the parent application. You can, however, perform a "save-as" operation to save a new private version of the reference in any application to which you have at least Read privilege.

Write Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned Write privilege for an object.

Object	Description
Cube	Members of the group can view and edit cube data, and can create private views of the cube. Write access does not allow you to edit data identified by consolidated elements or derived from rules. By definition, values derived by consolidation or by rules cannot be edited.
Element	Members of the group can view and edit data identified by the element.

Object	Description
Dimension	Members of the group can edit element attributes, edit element formats, and create private subsets for the dimension. Members of the group can also edit attributes for the dimension itself.

Reserve Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned Reserve privilege for an object.

Note that when you reserve an object, that reservation expires when the server containing the object shuts down.

Object	Description
Cube	Members of the group can view and edit data in the cube, and can reserve the cube to prevent other clients from editing cube data. You can release a cube you have reserved.
Element	Members of the group can view and edit data identified by the element, and can reserve the element to prevent other users from editing data. You can release an element you have reserved.
Dimension	Members of the group can add, remove, and reorder elements in the dimension, and can reserve the dimension to prevent other users from editing the dimension structure. You can release a dimension you have reserved.

Lock Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned Lock privilege for an object.

Note that there is no Unlock privilege, and that only users with Admin privilege for an object can unlock that object.

Object	Description
Cube	Members of the group can view and edit data in the cube, and can lock the cube. When a cube is locked, nobody can update its data.

Object	Description
Element	Members of the group can view and edit data identified by the element, and can lock the element. When an element is locked, nobody can update data identified by the element.
Dimension	Members of the group can add, remove, and reorder elements in the dimension, and can lock the dimension to prevent other users from editing the dimension structure. When a dimension is locked, nobody can edit the dimension structure.

Admin Privilege

The following table describes the ability of TM1 user groups to access various TM1 objects when assigned Admin privilege for an object.

Object	Description
Cube	Members of the group can read, write, reserve, release, lock, unlock, and delete the cube. They can also grant access privileges for this cube to other users.
Element	Members of the group can view, update, and delete cells identified by the element. They can reserve, release, lock, and unlock the element. They can also grant access privileges for this element to other users.
Dimension	Members of the group can add, remove, and reorder elements in the dimension. They can reserve, release, lock, and unlock the dimension. They can also create public subsets for the dimension and grant access privileges for the dimension to other users.
Application	Members of the group can see the application, use references within the application, and create both public and private references in the application. They can also create both public and private sub-applications. 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.

Object	Description
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.

Select Dimension

When you access the Security Assignment dialog box from an individual dimension, the Select Dimension option is available. This option lets you assign access privileges for elements in multiple dimensions.

After you assign access privileges for one dimension, click Save then select a new dimension from the Select Dimension list. When you complete assigning privileges for all desired dimensions, click OK to dismiss the dialog box.

Select Cube Dialog Box

Select the cube name you want to insert into your worksheet or formula and click **OK**.

Select Cube for Rules Dialog Box

Select the cube for which you want to create a new rule and click **OK**.

Select Dimension Dialog Box

Select the dimension name you want to insert into your worksheet or formula and click **OK**.

Select Dimension Worksheet Dialog Box

Select the dimension worksheet you want to open and click **OK**.

Select Element Dialog Box

Select the element name you want to insert into your worksheet or formula and click **OK**.

Select Rule Worksheet Dialog Box

Select the select the rule worksheet you want to open and click **OK**.

Server Explorer (Main Window)

Left pane (Tree pane)

Displays a hierarchical representation of all objects on servers to which you are currently connected.

Right pane (Properties pane)

Displays the properties of the object selected in the left pane of the Server Explorer. Properties vary according to the object selected.

File Menu

The following options are available on the File Menu in the Server Explorer.

Menu Item	Description
Options	Opens the TM1 Options dialog box.
Shutdown local server	Shuts down the local server and prompts you to save changes to data. This option is available only when the local server is running.
Start local server	Starts the local server. This option is available only when the local server is not running.
Refresh Available Servers	Updates the display of available servers in the left pane of the Server Explorer.
Exit	Closes the Server Explorer and any other windows associated with TM1 Perspectives/TM1 Architect.

Dynamic Menu

The options available from the second menu in the Server Explorer vary according to the type of object currently selected.

Servers Group

The following options are available from the TM1 menu when you select the servers Group in the Server Explorer.

Option	Description
Save Data All	Saves data on all servers to which you are currently connected.

Server

The following options are available from the Server Menu when you select an individual server in the Server Explorer.

Option	Description
Save Data	Saves all edits to data on the selected server.

Option	Description
Recycle (Clear memory for Local Server)	Shuts down and restarts the local server. When choosing this option you have the choice of recycling and saving data on the local server, or recycling and abandoning changes on the local server.
Shutdown	Shuts down the local server. This option is available only when the local server is selected.
Security, Clients/Groups	Opens the Clients/Groups Editor for the selected server. You must have Admin privileges for the server to access the Clients/Groups Editor.
Security, Change Password	Opens the Password Change dialog box, from which you can change your password on the selected server.
Security, Refresh Security	Update all security structures/assignments on the selected server.

Option	Description
Capability Assignments	Allows the administrator to set permissions for specific features by usergroup. At the intersection of the usergroup and the capability, administrators can set Grant or Deny (same as blank) to enable or disable that capability. Some capability settings may be ignored depending on the configuration settings made on the server. The following capabilities can be set per usergroup: • **Block Access to Server Explorer** To prevent the Server Explorer from launching, click the interesection of this capability and the usergroup and select **Grant**. Blank or Deny means the Server Explorer is used by this usergroup. • **Personal Workspace Writeback Mode** To enable a usergroup to use Personal Workspaces, click the interesection of the usergroup and this capability and select **Grant**. Blank or Deny means this usergroup does not use Personal Workspaces. If DisableSandboxing is set to T, this capability assignment is ignored. • **Sandbox** To enable a usergroup to use Sandboxes to create multiple what-if scenarios, click the interesection of the usergroup and this capability and select **Grant**. Blank or Deny means this usergroup cannot use multiple Sandboxes. If DisableSandboxing is set to T, this capability assignment is ignored. See the IBM Cognos TM1 *Operation* and *User Guides* for more information.
View Transaction Log	Opens the Transaction Log Query dialog box, from which you can view a log of transactions on the selected server.
View Message Log	Opens the Message Log dialog box, which displays messages recorded on the selected server.
Start Performance Monitor	Initiates performance monitoring. When the Performance Monitor is running TM1 populates several control cubes that let you track statistics for cubes, clients, and server.
Stop Performance Monitor	Stops performance monitoring.

Option	Description
Deferred Updates, Start Batch Updates	Starts batching updates to be sent to the selected server.
Deferred Updates, End Batch Updates	Ends batching updates and sends all edits to the selected server.
Server Manager	Opens the Clients Messaging Center dialog box, from which you can shutdown the selected server, disconnect clients, and broadcast messages.
Cancel Shutdown	Cancels a previously executed server shutdown.
Disconnect Self	Disconnects your client from the selected server.
Who Am I	Returns a message indicating your user name on the server.

Applications

The following options are available from the Applications Menu when you select either the Applications group or an individual application in the Server Explorer.

Option	Description
Open	Expands the selected application or Applications group to reveal references and sub-applications.
Close	Collapses the selected application or Applications group to hide references and sub-applications.
Delete	Deletes the selected application. When you delete an application, all sub-applications and references within the application are automatically deleted. This option is not available when the Applications group is selected.
Rename	Sets the selected application name in edit mode, so you can type a new name for the application. This option is not available when the Applications group is selected.
Security, Security Assignments	Opens the TM1 Security Assignments window, from which you can assign security privileges for the references and immediate sub-applications contained within the selected application or Applications group.

Option	Description
Security, Make Public	Choose this option to publish a private application. When you publish an application, all sub-applications and private references to public objects within the application are automatically published as well. This option is not available when the Applications group is selected.
Security, Make Private	Choose this option to privatize a public application. When you privatize an application, all sub-applications and public references within the application are automatically privatized as well. This option is not available when the Applications group is selected.

Cubes

The following options are available from the Cubes Menu when you select a cubes group in the Server Explorer.

Option	Description
Create New Cube	Opens the Creating Cube dialog box.
Edit Attributes	Opens the Attributes Editor for the selected cube.
Security Assignments	Opens the TM1 Security Assignments dialog box for the cubes in the selected cube group. You must be a member of the Admin group on the server containing the cube group to access this dialog box.

Cube

The following options are available from the Cube Menu when you select a cube in the Server Explorer.

Option	Description
Browse	Opens the cube for browsing in the Cube Viewer window.
Browse in Excel	Opens the cube for browsing in the In-Spreadsheet Browser.
Pick	Copies the cube name to the Clipboard.
Create New Cube	Opens the Creating Cube dialog box.
Unload Cube	Unload the selected cube from the server's memory.

Option	Description
Delete Cube	Deletes the selected cube and all associated data. You must have Admin privileges to delete a cube
Re-order Dimensions	Opens the Cube Optimizer window, from which you can optimize the order of dimensions in the selected cube.
Create Rule	Opens the Rules Editor, from which you can create a rule for the selected cube.
Delete Rule	Deletes the rule associated with the selected cube. You must have Admin privileges for a cube to delete the associated rule.
Export as ASCII Data	Exports the data contained in the selected cube to a comma-delimited (.cma) ASCII file.
Synchronize Data	Synchronizes the data in the selected cube with data from the associated replication server.
Security, Reserve	Temporarily reserves the selected cube so that other clients cannot edit data in the cube. You must have Reserve privileges to reserve a cube.
Security, Release	Releases a cube you have reserved so that other clients can edit data in the cube. You must have Reserve privileges to release a cube.
Security, Lock	Permanently locks the selected cube so that other clients cannot edit data in the cube. The client you are logged in with also becomes locked out of these elements. You must have Lock privileges to lock a cube.
Security, Unlock	Unlocks the selected cube so that other clients can edit data. You must have Admin privileges to unlock a cube.
Properties	Opens the Cube Properties dialog box, from which you can set measure and time dimensions.

Dimensions

The following options are available from the Dimensions Menu when you select a dimensions group in the Server Explorer.

Option	Description
Create New Dimension	Opens the Dimension Editor window, from which you can create a new dimension.
Edit Attributes	Opens the Attributes Editor window, from which you can assign and edit attributes for all dimensions in the selected group.
Security Assignments	Opens the TM1 Security Assignments dialog box, from which you can assign security privileges for each dimension in the group. You must be a member of the Admin group to use this option.

Dimension

The following options are available from the Dimension Menu when you select a dimension in the Server Explorer.

Option	Description
Insert New Subset	Opens the Subset Editor window for the dimension.
Pick	Copies the dimension name to the Clipboard.
Edit Dimension Structure	Opens the selected dimension for editing in the Dimension Editor window. You must have Write privileges for the selected dimension to use this option.
Create New Dimension	Opens an empty Dimension Editor window, from which you can create a new dimension. You must be a member of the Admin group to create a new dimension.
Export Dimension	Exports the selected dimensions as a comma-delimited (.cma) file.
Delete Dimension	Deletes the selected dimension. You must be a member of the Admin group to delete a dimension.
Set Elements Order	Opens the Dimension Element Ordering dialog box, from which you can set the order of elements in the selected dimension.
Edit Element Attributes	Opens the Attributes Editor window, from which you can assign and edit attributes for all elements in the selected dimension.

Option	Description
Synchronize Data	Synchronizes the data in the selected dimension with associated data from any replicated servers.
Security, Reserve	Temporarily reserves the selected dimension so that other clients cannot edit the dimension structure. You must have Reserve privileges to reserve a dimension. Note that this option reserves only the dimension structure. It does not reserve any data identified by elements in the selected dimension.
Security, Release	Releases a reserved dimension so that other clients can edit the dimension structure. You must have Reserve privileges to release a dimension. Note that this option releases only the dimension structure. It does not release any data identified by elements in the selected dimension.
Security, Lock	Permanently locks the selected dimension so that other clients cannot edit the dimension structure. You must have Lock privileges to lock a dimension. Note that this option locks only the dimension structure. It does not lock any data identified by elements in the selected dimension.
Security, Unlock	Unlocks the selected dimension so that other clients can edit the dimension structure. You must have Admin privileges to unlock a dimension. Note that this option unlocks only the dimension structure. It does not unlock any data identified by elements in the selected dimension.
Security, Elements Security Assignments	Opens the TM1 Security Assignments dialog box, from which you can assign security privileges for each element in the dimension. You must have Write privileges for the selected dimension to use this option.

CubeViews

The following options are available from the CubeViews Menu when you select a views group in the Server Explorer.

Option	Description
Create New View	Opens the Cube Viewer window, from which you can configure a new view.

CubeView

The following options are available from the CubeView Menu when you select a view in the Server Explorer.

Option	Description
Browse	Opens the view in the Cube Viewer window.
Browse in Excel	Opens the view in the In-Spreadsheet Browser.
Export as Text Data	Opens the View Extract window, from which you can export the view as a comma-delimited (.cma) file.
Publish	This option is available when you select a private view. Choose this option to convert a view from private to public. Public views are available to all clients with Read privileges for the cube containing the view.
Delete View	Deletes the selected view. Note that this option only deletes the view configuration, and not the data contained in the view.

Subsets

The following options are available from the Subsets Menu when you select a subsets group in the Server Explorer.

Option	Description
Insert New Subset	Opens the Subset Editor window, from which you can define a new subset.

Subset

The following options are available from the Subset Menu when you select a subset in the Server Explorer.

Option	Description
Open	Opens the selected subset in the Subset Editor window.
Create New Subset	Opens the Subset Editor window for the dimension to which the selected subset belongs. You can define a new subset in this window
Publish	This option is available when you select a private subset. Choose this option to convert a subset from private to public. Public subsets are available to all clients with Read privileges for the dimension containing the subset.
Delete Subset	Deletes the selected subset. Note that this option only deletes the subset configuration, and does not delete the elements contained in the subset from the parent dimension.

Replications

The following options are available from the Replications Menu when you select a replications group in the Server Explorer.

Option	Description
Insert New Replication	Opens the Create Server Replication Object dialog box, from which you can establish a new replication connection.

Replication

The following options are available from the Replication Menu when you select a replication in the Server Explorer.

Option	Description
Synchronize Data	Synchronizes data between the target and source servers.
Modify Replication Parameters	Opens the Create Server Replication Object dialog box, from which you can modify the parameters for the selected replication connection.
Delete Replication	Deletes the selected replication connection.
Display Chores Involved	Opens the Select Chores to Modify dialog box. You can use this dialog box to remove the selected replication from any associated chores.

Replicated Cube

The following options are available from the Cube Menu when you select a replicated cube in the Server Explorer.

Option	Description
Replicate	Opens the Replicate Cube dialog box for the selected cube, from which you can define replication parameters and replicate the cube.
Synchronize Data	Synchronizes data between the replicated cube and the source server.

Processes

The following options are available from the Processes Menu when you select a processes group in the Server Explorer.

Option	Description
Create New Process	Opens TurboIntegrator, from which you can create a new process.
Security Assignments	Opens the TM1 Security Assignments dialog box, from which you can set security privileges for processes on the current server.

Process

The following options are available from the Process Menu when you select a process in the Server Explorer.

Option	Description
Display Chores Involved	Opens the Select Chores to Modify dialog box. You can use this dialog box to remove the selected process from any associated chores.
Edit Process	Opens the selected process in a TurboIntegrator window.
Run Process	Runs the selected process.
View	Views a process in read-only mode. Allows members of the DataAdmin and SecurityAdmin groups to view a process in read-only mode when the Security Access option is enabled for the process.
Security Access	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. You set this option on a process-by-process basis. For details, see the section about securing processes in the IBM Cognos TM1 *Developer Guide*.
Delete Process	Deletes the selected process.
Use Active Sandbox	Configures the process to use the data in the current active sandbox instead of base data when you run the process. The active sandbox is determined by which sandbox is currently selected in the Cube Viewer.

Chores

The following options are available from the Chores Menu when you select a chores group in the Server Explorer.

Option	Description
Create New Chore	Opens the Chore Setup Wizard, from which you can schedule a new chore.
Security Assignments	Opens the TM1 Security Assignments dialog box, from which you can set security privileges for chores on the current server.

Chore

The following options are available from the Chore Menu when you select an individual chore in the Server Explorer.

Option	Description
Activate Schedule	This option toggles the chores execution status. Select this option to activate the selected chore for execution. A check mark displays next to this option when a chore is activated. Select this option again to deactivate the selected chore.
Edit	Opens the chore for editing in the Chore SetUp Wizard. You must deactivate a chore before editing.
Run	Runs the selected chore.
Delete	Deletes the selected chore. You must deactivate a chore before deleting.

Edit Menu

The following options are available on the Edit Menu in the Server Explorer.

Option	Description
Copy	Copies the selected object label to the Clipboard.
Delete	Deletes the selected object from the server.

View Menu

The following options are available on the View Menu in the Server Explorer.

Option	Description
Status Bar	Hides or displays the status bar at the bottom of the Server Explorer window. A check mark indicates that the status bar is displayed.
Toolbar	Hides or displays the toolbar at the top of the Server Explorer window. A check mark indicates that the toolbar is displayed.
Properties Window	Hides or displays the Properties pane of the Server Explorer. A check mark indicates that the Properties pane is displayed.

Option	Description
Objects: Applications Cubes Dimensions Replications Processes Chores	Hides or displays any of the objects in the Server Explorer's left pane (Tree pane). A check mark indicates that the selected object is displayed.
Collapse All Children	Contracts the tree in the left pane of the Server Explorer to hide all children of a selected object.
Expand All Children	Expands the tree in the left pane of the Server Explorer to show all children of a selected object.
Display Control Objects	Hides or displays the control cubes and dimensions in the left pane of the Server Explorer window. A check mark indicates that the control objects are displayed.
Refresh	Updates the current hierarchical display of objects in the left pane of the Server Explorer.

Subset Editor

Elements pane

Displays a hierarchical representation of all elements in the subset you are currently viewing.

Properties pane

Displays the properties of the elements selected in the Elements pane of the Subset Editor. When you select a consolidated element, this pane displays the names, types, and weights of all children of the consolidated element.

Note: When viewing an exceptionally large dimension set in the Subset Editor with the Properties pane on, you might experience performance issues. This can happen when you select a consolidation in the Elements pane and TM1 has to display the entire list of related elements and properties in the Properties pane.

If you are working with large dimension sets, you may want to turn off the Properties pane. To turn off the Properties pane, click the Properties Window option in the View Menu to remove the check mark next to the option.

Subset Menu

Menu Item	Description
Open	Opens the TM1 Save Subset dialog box. Select a subset from the list and click OK to open the subset.
Reload	Reloads the current subset definition.
Save	Saves the current subset definition.
Save as	Saves the current subset definition under a new name.
Close	Closes the Subset Editor.

Edit Menu

Menu Item	Description
Undo	Undoes last action.
Redo	Restores the last "undo" action.
Cut	Cuts selected elements to the Clipboard.
Copy	Copies selected elements to the Clipboard.
Copy Unique Name	Copies the element name, as an MDX expression, to the Clipboard. The copied element name can then be pasted into the Expression Window of the Subset Editor.
Paste	Pastes the contents of the Clipboard at the current insertion point.
Paste Above	Paste the contents of the Clipboard above the currently selected element.
Paste Below	Paste the contents of the Clipboard below the currently selected element.
Insert Subset	Opens a new instance of the Subset Editor so you can add a user-defined consolidation to the current subset.
Keep	Keeps only the currently selected elements in the Elements pane of the Subset Editor, and removes all other elements.
Delete	Removes selected elements from the current subset definition.

Menu Item	Description
Pick Elements, Horizontal	Copies selected elements to the Clipboard in a horizontal orientation, so they can be pasted into a worksheet row.
Pick Elements, Vertical	Copies selected elements to the Clipboard in a vertical orientation, so they can be pasted into a worksheet column.
Sort, Descending	Sorts all elements in the Elements pane in alphabetically descending order.
Sort, Ascending	Sorts all elements in the Elements pane in alphabetically ascending order.
Sort, Hierarchy	Sorts all elements in the Elements pane in hierarchical order, so you can see the parent/child relationship of elements.
Sort, Index Ascending	Sorts all elements in the Elements pane in ascending order according to element index value.
Sort, Index Descending	Sorts all elements in the Elements pane in descending order according to element index value.
Drill Down	Displays the immediate children of selected elements.
Roll Up	Displays the immediate parents of selected elements.
Expand Element	Displays all children of selected elements.
Collapse Element	Collapses selected consolidations so that children are not displayed.
Filter by, Levels	Opens the Filter by Level dialog box, from which you can select elements by hierarchy level.
Filter by, Attribute	Opens the Filter by Attribute dialog box, from which you can select elements by attribute value.
FIlter by, View Extract	Lets you select only those elements that satisfy a user-defined query. This option is available only when you open the Subset Editor by clicking on a dimension label in the Cube Viewer window.
Filter by, Wildcard	Lets you select elements that match a user-defined search string.

Menu Item	Description
Select Alias	Opens the TM1 Aliases dialog box, from which you can select a previously defined alias by which to display element names.
Security, Reserve	Temporarily reserves the selected element so that other clients cannot edit data identified by the element. You must have Reserve privileges to reserve an element.
Security, Release	Releases a reserved element so that other clients can edit data identified by the element. You must have Reserve privileges to release an element.
Security, Lock	Permanently locks the selected element so that other clients cannot edit data identified by the element. You must have Lock privileges to lock an element.
Security, Unlock	Unlocks the selected element so that other clients can edit data identified by the element. You must have Admin privileges to unlock a dimension.
Edit Element Formats	Opens the Edit Element Formats worksheet, where you can define display styles for dynamic slices and TM1 Websheets.

View Menu

Menu Item	Description
Toolbars	Opens a submenu that lets you enable or disable the display of all Subset Editor toolbars. A check mark indicates that a toolbar is displayed.
Status Bar	Hides or displays the Status Bar at the bottom of the Subset Editor window. A check mark indicates that the Status Bar is displayed.
Properties Window	Hides or displays the Properties pane. A check mark indicates that the Properties pane is displayed.

Menu Item	Description
Expression Window	Hides or displays the Expression Window at the bottom of the Subset Editor. A check mark indicates that the Expression Window is displayed.
Expand Above	This option determines how consolidations expand and contract when you drill down. When this option is turned on, children of a consolidation expand above the consolidation when you drill down. When this option is turned off, children of a consolidation expand below the consolidation when you drill down. When the Expand Above option is enabled in a subset, drilling down on a consolidation in either the Cube Viewer, In-Spreadsheet Browser, or slice results in the following behavior: If the option is enabled in a row subset, drilling down on a consolidation displays the children above the consolidation. If the option is enabled in a column subset, drilling down on a consolidation displays the children to the left of the consolidation.
Refresh	Updates the display of the Elements pane.

Tools Menu

Menu Item	Description
Record Expression	Starts recording your actions in the Subset Editor.
Stop Recording	Stops recording your actions in the Subset Editor. When you stop recording, TM1 generates an MDX expression that can be saved to create a dynamic subset.
Clear Expression	Clears the contents of the Expression Window.
Filter	Opens the Filter Subset dialog box, which lets you create a dynamic subset based on cube values.

Aliases Dialog Box

To view current subset elements by assigned aliases, select an alias name from the list and click **OK**.

TM1 Options Dialog Box

The following options can be set in the TM1 Options dialog box.

Login Parameters

Option	Description
Admin Host	Enter the computer name of your Admin Host. The Admin Host is the computer on which your Admin Server runs.
Integrated Login	Toggle this option on to use Integrated Login. Toggle this option off to use standard TM1 login security. The default is off.

Local Server

Option	Description
Local Server Data Directory	Enter the full path to your Local Server Data Directory, or click the accompanying Browse button to browse to the directory. You can also click the down arrow to select from a list of recently accessed directories.
Connect to Local Server on Startup	Toggle this option off to start TM1 Perspectives/TM1 Architect without launching the local server. The default is on.

Note: The local server is supported only on 32-bit versions of TM1. The default data directory for the local server is Pdata. If you are running a 64-bit version of TM1, the Sdata sample server, which is installed by default with the TM1 server, contains the same objects and data as are found in Pdata.

Admin Server Secure Socket Layer

Option	Description
Certificate Authority	The full path of the certificate authority file that issued the Admin Server's certificate.

Option	Description
Certificate Revocation List	The full path of the certificate revocation file issued by the certificate authority that originally issued the Admin Server's certificate. A certificate revocation file will only exist in the event a certificate had been revoked.
Certificated ID	The name of the principal to whom the Admin Server's certificate is issued.
Use Certificate Store	Select this option if you want the certificate authority certificate which originally issued the Admin Server's certificate to be exported from the Windows certificate store at runtime. When this option is selected, you must also set a value for Export Certificate ID in the TM1 Options dialog box.
Export Certificate ID	The identity key used to export the certificate authority certificate, which originally issued the Admin Server's certificate, from the certificate store. This parameter is required only if you enable the Use Certificate Store option.

Transaction Log Query Dialog Box

The Transaction Log Query dialog box lets you query and view records in the TM1 transaction log (Tm1s.log). The dialog box contains fields for four parameters that you must specify to execute a query.

Option	Description
Start Time	The start date/time for the query. TM1 queries against all records written to the transaction log on or after this date/time. You must use the format MM/DD/YYYY HH:MM:SS to specify a start time. The default start date/time is 00:01:00 GMT on the date the query is launched.
End Time	The end date/time for the query. The default is __/__/____ __:__:__, which is an open-end date/time. If you accept the default, TM1 queries against all records up to the time the query is launched.

Option	Description
Client(s)	The client(s) against which the query is applied. You can query against either a single client or all clients. The default is all clients (*).
Cubes(s)	The cube(s) against which the query is applied. You can query against either a single cube or all cubes. The default is all cubes (*).

To set any of the above parameters, click the arrow next to the appropriate field.

Transaction Log Query Results Dialog Box

The Transaction Log Query Results dialog box presents the result of a transaction log query in table format. The table contains the following columns for each record returned by the query:

Column	Description
LOGTIME	The time at which a value was edited.
REPLICATIONTIME	The time at which a value was replicated.
CLIENT	The name of the client who wrote the value.
OLDVALUE	Data value before editing.
NEWVALUE	Data value after editing.
CUBENAME	The cube to which the value was written.
KEY N	There are multiple Key N columns in the table, each column representing the elements that identify the value.

The Transaction Log Query Results dialog box includes three menus.

The **File** Menu contains a single item: Exit.

The **Help** Menu contains a single item to open help for the dialog box.

The **Edit** Menu contains the following items:

Menu Item	Description
Copy	Copies a single selected cell to the clipboard.

Menu Item	Description
Hide	Suppresses the display of selected record(s) in the table. You can click Refresh to restore the display of hidden records.
Sort	Opens a sub-menu from which you can choose columns to sort or a sort order to apply.
Find	Opens the Find/Replace dialog box, which allows you to search the current table.
Select	Selects highlighted record(s)
Unselect	Unselects highlighted record(s).
Select All	Selects all records in the table.
Unselect All	Unselects all records in the table.
Back Out	Backs out selected record(s). When a record is backed out, the OLDVALUE for the record replaces the NEWVALUE for the record. When multiple records for a single cube location are selected, records are backed out to OLDVALUE of the earliest LOGTIME.

TurboIntegrator Editor

The TurboIntegrator Editor lets you define processes for importing data or metadata from several possible sources. The editor is comprised of five tabs, several of which are dynamic or contain sub-tabs. You define a process by completing each tab in sequential order.

File Menu

Menu Item	Description
Save	Saves the current process definition.
Save As	Saves the current process definition with a new name.
Run	Runs the current process.
Exit	Closes the TurboIntegrator Editor.

Edit Menu

Menu Item	Description
Undo	Undoes the last typing action that was performed on the Prolog, Metadata, Data, or Epilog procedure sub-tab.
Cut	Cuts the selected text to the Clipboard.
Copy	Copies the selected text to the Clipboard.
Paste	Pastes the contents of the Clipboard to the current field or cell.

Data Source Tab

Use the Data Source tab to identify and access the source from which you want to import data.

Note: When defining a process from the TM1 client, the path to an ASCII or ODBC data source may differ from the path used by the server. If this happens, the process will fail. To ensure that your processes work correctly:

- Define processes involving ODBC data sources on the actual server where the process is to reside. Do not use a remote client to define such a process.
- Use the Windows Network Neighborhood to define the path to ASCII data sources. This ensures that the path is unambiguous to both clients and servers.

The fields and options available on the Data Source tab vary according to the Datasource Type you select. The following tables describes the required fields and options for each source.

ODBC

Define an ODBC datasource:

Fields	Description
Data Source Name	The full path to the ODBC data source.
UserName	Your user name on the source.
Password	Your password.
Query	An SQL query to extract data from the source.
Use Unicode	Check here to use Unicode for this source.
Preview	Displays the first 10 records.

Text

Define an ASCII or Text datasource:

Fields	Description
Data Source Name	The full path to the source text file. To ensure that this path is recognizable to both client and server, click the Browse button and use the Network Neighborhood to define the path.

Fields	Description
Data Source Name On Server	When you create a new process, TurboIntegrator assumes that the data source name on the server is identical to the data source name used to create the process. If the data source name on the server is different from the local data source used to create the process, enter the full path to the data source file on the server.
Delimiter Type	If the source uses a character to define the columns, select **Delimited**, then choose the character in the **Delimiter** box.
Fixed Width	If the source uses a fixed width, select **Fixed Width**, then use the **Set Field Widths** button to open the Preview dialog box to set column widths.
Quote Char	Specify the quote character used in your source data.
Number of title records	If the title records span more than one row, enter the number of rows here. Otherwise, leave this field blank.
Number Delimiters	Enter the character to use for the Decimal **Separator** and **Thousand** Separator in the source.

ODBO

Define an ODBC data source established for a relational database that can be built using the Microsoft Windows Data Sources control panel:

Cube or Dimension:

Select the type of import to do:

Fields	Description
ODBO Provider	To import a ODBO cube, use the pull-down to select the OLE DB for OLAP data source, including Microsoft Analysis Services to use.
ODBO Location	Enter the file location for this provider. When you import a dimension, leave this field blank.
ODBO Datasource	Enter the name your administrator assigns to a set of catalogs at a particular location. In Microsoft Analysis Services, this is the name of a registered server.
ODBO Catalog	The name assigned by your administrator to a particular collection of databases (Cubes, Dimensions and other objects). For MAS, this is the name of the database.

Fields	Description
ODBO UserID	A valid username for the database.
ODBO Password	A valid password in this datasource.
Additional Connection Parameters	Enter any additional parameters required by the ODBO source here. Delimit the parameters with semi-colons. When you import a dimension, leave this blank.

MDX Query:

Define an MDX-based query:

Fields	Description
Connection tab	Enter the required connection information. Connection parameters are vendor-specific. Click Connect. If you connect successfully, the MDX Query tab becomes available.
MDX Query tab	Enter the MDX query into this tab or paste a working MDX query from another application.

SAP

Defines the SAP RFC datasource:

Tab	Field	Description
Connection	System	The name of the SAP system you want to connect to. If the system name includes spaces, enclose the name in double quotes.
	Client	A number that corresponds to the UI version on the SAP server. For example, 498.
	User	Your username on the SAP system.
	Password	Your password on the SAP system.

Tab	Field	Description
	Language	The language you want to use to logon to the SAP system. All textual descriptions are returned in the language specified, if available. The language parameter is a two-letter abbreviation, for example, EN=English.
	Additional Connection Parameters	Enter any other parameters and values you use to connect to your SAP BW system.
	Packet Size	A value that limits the number of rows in each packet sent from SAP to TM1 . A smaller packet size will result in increased network traffic with small packets, while a larger packet size results in decreased network traffic but larger packets per transmission. The default packet size, which is also the minimum packet size, is 50,000.

Info Cube

Area	Field	Description
Info Cube	Show SAP Technical Names	To use technical names, select this checkbox. Leave this box unchecked to display by descriptive name.
	Select InfoCube to Load from	Use the option to indicate the InfoCube from which you want to import data.
	Select TM1 Cube to Load to	To import the SAP InfoCube to an existing TM1 cube, click this option and select the cube to receive the SAP InfoCube data.
	Select TM1 Cube to Load to	To create a new TM1 cube when you import the InfoCube, enter a name for the new TM1 cube in this to field.

Area	Field	Description
TM1 Cube Action	Create	Imports data and metadata from the SAP InfoCube and creates a new cube in TM1 . Use this option only when none of the cubes and dimensions you are importing exist on the server.
	Recreate	Destroys an existing TM1 cube and rebuilds it using data and metadata from the SAP InfoCube. Use this option only when the TM1 cube and dimensions exist, and you want to replace them with new structures and data from the SAP InfoCube.
	Update	Imports data from an existing SAP InfoCube cube and inserts it into an existing TM1 cube. This option does not change the structure of cubes and dimensions on the server.
		Processes that specify No Action do not affect the data or metadata of TM1 cubes. Use this option to test and debug processes or to define your own custom operations.
Data Action	Store Values	This option writes cell values from the SAP InfoCube to the TM1 cube. If you choose this option when the Update Cube option is selected, existing TM1 cube values are overwritten by values imported from the InfoCube.
	Accumulate Values	The Accumulate Values option allows you to aggregate existing TM1 Cube values with values imported from the SAP InfoCube.

Area	Field	Description
	Zero Out Portion of Target Cube	This option becomes available when you select the Update Cube action. Select this option if you want to set all data points in a specified cube view to zero. To define the cube view to be zeroed, you can: • Click the View list to select an existing view to be zeroed. • Click the **More** button next to the View option list to define a new view to be zeroed.
	Enable Cube Logging	To log changes to cube data while importing from an SAP InfoCube, select this option. To disable logging while importing, clear this option. **Note:** Disabling logging accelerates data loading and updating, but makes it impossible to recover any updates in the event of a system failure.

Characteristics tab

Field	Description	
Select Hierarchies	Identify the hierarchies in the datasource.	
Evaluation Date	Date when all time-dependent SAP attributes are imported into TM1 as they existed on the specified date. Attributes that are not time-dependent are imported as they exist at the time of process execution. If this date is cleared, *all* SAP attributes are imported as they exist on the date the TM1 process runs. Do not import a hierarchy with intervals.	
TM1 Dimension	Select the existing TM1 dimension that maps to this characteristic. Leave this field empty if you do not want to import the characteristic in to your TM1 cube.	
TM1 Dimension Action	Create	Create a new TM1 dimension from the SAP characteristic.

Field	Description	
	Recreate	Entirely recreate an existing TM1 dimension with elements imported from the SAP characteristic.
	Update	Update an existing dimension structure by adding new elements imported from the SAP characteristic.
	AsIs	Process the characteristic through TurboIntegrator, but do not use the characteristic to create or modify any TM1 dimensions. Use this option to test and debug processes or to manipulate the characteristic in the Advanced tab of TurboIntegrator.
	Don't Load	Do not import the SAP characteristic into TM1 . The characteristic is entirely excluded when the SAP InfoCube is processed through TurboIntegrator.
Select Attributes	Characteristic Attributes	Define the attributes for this data source.
	Text	Identifies attributes with a string value.
	Numeric	Identifies attributes with a numeric value.
	Alias	Identifies attributes that are alternative names for the dimensions with which they are associated. A dimension alias must be unique from all other dimension aliases or actual dimension names
Select Key Figure	Select each key figure you want to import into TM1 . If the key figures map to an existing TM1 dimension, click the **TM1** Dimension column and select the dimension that corresponds to the key figures.	
Restrictions	Add Restrictions	Create a new restriction for this characteristic.

Field	Description	
	SAP Characteristic	Select the characteristic to set a restriction on.
	Sign	Indicates if the restriction is inclusive or exclusive. Choose Include if you want the TurboIntegrator process to import only those values that fall within the restriction definition. Choose Exclude if you want the TurboIntegrator process import only those values that fall outside of the restriction definition.
	Option	The Operator used for the restriction. There are eight operators to choose from, as described in the following table.

Option Restriction Operators

Operator	Description
=	The restriction identifies only characteristics equal to the specified Low Value.
< >	The restriction identifies only characteristics less than or greater than the specified Low Value.
<	The restriction identifies only characteristics less than the specified Low Value.
>	The restriction identifies only characteristics greater than the specified Low Value.
< =	The restriction identifies only characteristics less than or equal to the specified Low Value.
> =	The restriction identifies only characteristics greater than or equal to the specified Low Value.
[ ]	The restriction identifies only characteristics that fall between the specified Low Value and High Value, inclusive.

Operator	Description
] [	The restriction identifies only characteristics that fall outside of the specified Low Value and High Value, inclusive.

There are eight operators to choose from, as described in the following table.

Enter a low value for the restriction in the Low Value column.

Enter a high value for the restriction, if required, in the High Value column.

Note: Restrictions are not validated through TurboIntegrator. You must ensure that the restrictions you enter are accurate and valid for your SAP data.

Security

Field	Description
Import Security	Indicates that the security assignments for this characteristic should be imported.
Top Consolidation	Creates a top-level consolidation for the TM1 dimension created using the name entered here.

Field	Description
Make Texts Unique	To generate unique aliases for all elements created from the SAP characteristic, select this option. When you import an SAP characteristic into TM1 , characteristic values become TM1 dimension elements while SAP value descriptions become TM1 element aliases. In TM1 , all element aliases within a dimension must be unique. If a TurboIntegrator process attempts to assign the same alias to multiple elements, the process will generate errors and alias creation will fail. When Make Texts Unique option is selected, TM1 examines the SAP descriptions that are imported and converted into TM1 aliases. If TM1 detects that multiple values use identical descriptions, TM1 appends the value name to the description to generate unique aliases. In TM1 version 9.5.2 the V2 variable does not contain any values so the only way to make the Alias attribute import is using the **Make Texts Unique** option. If **Make Texts Unique** is not checked, no SAP_Text data is fetched. When **Make Text Unique** is not checked, you can add `DataSourceSAPUsingTexts=1;`in the prolog to import the alias attribute values. See the AttrPutS function to get the same behavior using a TI script.
Evaluation Date	All characteristic values that existed between the selected date and the date of process execution will be imported into TM1 . When there is no evaluation date specified, the default is the date on which the TurboIntegrator process is executed.

SAP Table

Field	Description
SAP Table	Indicates that the data source is an SAP table query.
Table Name	Name of the SAP table to use.

Field	Description
Filter string	An SQL filter string to be used in the WHERE clause when the SQL SELECT statement that is generated by TurboIntegrator is executed against the SAP table.

ODS Table

Field	Description
ODS Table	Used to export TM1 data to an ODS table which can then be used to import data through a SAP Infocube.
ODS Setup	Define the details of the ODS table.
Browse	Select the TM1 View to use as the data source.
Show Technical Names	To use technical names, select this checkbox. Leave this box unchecked to display by descriptive name.
Select ODS Table	Select the ODS table to export to.

Field	Description
Columns	Columns may be either SAP characteristics or key figures. You must be familiar with the structure of the ODS table to know which columns are characteristics and which are key figures; TurboIntegrator does not differentiate the ODS table column types. You should be aware of the following details when mapping dimensions to characteristics: • You do not have to map a dimension to every characteristic column in the ODS table. Some columns may not have a corresponding TM1 dimension when the mapping is complete. In this case, any characteristic column that is not mapped will be empty when the export is completed. • You should not map a single TM1 dimension to multiple ODS characteristic table columns. The TurboIntegrator user interface does not prevent you from doing so, but such mapping will result in redundant column values in the ODS table. • When you map a TM1 view title dimension to a characteristic, and the title dimension *does not* use a named subset, only the last element in the current unnamed title subset is exported to the ODS table. If the title dimension *does* use a named subset, all subset elements are exported to the ODS table.
Select Measure	The last dimension in the source cube view is assumed to be the measures dimension. When you map a measures dimension to an ODS table column, the Select Measure button becomes available. Select the single element that maps directly to the key figure column in the ODS table If your ODS table includes a single key figure column, you can also use the alternate key figure.

Field	Description
TM1 Dimension	If your ODS table includes a single key figure column, you can use the _TM1CellValue_ option to map TM1 cube values to the ODS table. You *cannot* use the _TM1CellValue_ option if your ODS table contains multiple key figure columns. To use this option, do not map the TM1 measures dimension to the key figure column. Instead, click the **TM1** Dimension column and select **_TM1CellValue_**. When you use this alternate method to map TM1 cube values, the TM1 measures dimension should *not* be mapped to any ODS column.

Currency

Field	Description
SAP Currency	Used to import currency data to a new or existing three-dimensional cube on your server.
Show SAP Technical Names	To use technical names, select this checkbox. Leave this box unchecked to display by descriptive name.
Enter Cube Name	Enter an existing three-dimensional cube or enter a new cube name.
From Currency	Select the initial currency to import from SAP. The list of available currencies reflects the currencies defined in your SAP system.
Target Dimension	Specify the TM1 dimension to receive the SAP currency strings.
To Currency	Select the second currency to import from SAP.
Conversion Type	Select the conversion method to use when converting the initial currency to the second currency.

Cognos TM1

Uses an IBM Cognos TM1 cube or dimension as the datasource.

Field	Description
Cube View Dimension Subset	Use the Browse button to select an available TM1 view or Dimension to use as the data source. Click the Preview button. Then complete the fields on the other tabs.

IBM Cognos Package Connector

Indicates that the datasource is a published IBM Cognos Package created from an SAP query.

Package

Field		Description
Connection	Define the connection to this data source.	
	Authentication Namespace	Displays all created IBM Cognos Namespaces currently available.
	UserID	Password
Package	Select Package	Click the Browse button to select an available publish Package.
	Select TM1 cube to load to	If you are importing the data directly into an existing TM1 cube, enter the cube name here or use the pull-down.
	Data Action, Cube Action, Enable Cube Logging	See the descriptions of these fields in the SAP Info Cube above.

Dimension

Field	Description	
Dimension	Package	Identify the Package to use for this dimension.
	Dimension to load from	Identify the dimension to use.
	Dimension to load into	Identify the dimension to import into.

Field	Description	
	TM1 Dimension Action	See the description for the Characteristics tab for details.
	Retrieve Security Settings	Use the security on the dimension.
	Top Consolidation	The name of a top-level consolidation for the TM1 dimension with all imported elements as children of the consolidation.
	Select Hierarchies	Select and map the hierarchies to use in this import and define how they are mapped into the new TM1 dimension. See the IBM Cognos TM1 *TurboIntegrator Guide* for details.
	Select Attributes	Select the Attributes to use and define the mapping. See the IBM Cognos TM1 *TurboIntegrator Guide* for details.

None

Used to add a user-defined prolog to a process.

If the data source for the process is None, TurboIntegrator immediately executes the Epilog procedure after the Prolog finishes processing.

Note: When the data source for a process is None, the Metadata and Data procedures are ignored. In this case, all scripts for the process must be created in either the Prolog or Epilog procedures.

Preview Grid

The preview grid displays the first ten records in your data source. Use this grid to confirm that the source is correct and to help determine the structure of records.

If you change your data source, click Preview again to refresh the display of the grid.

Variables Tab

The Variables tab includes a grid and two buttons.

Grid

Use the Variables grid to assign variables and identify the contents of each column in your data source. The Variables grid includes the following columns.

Column	Description
Column ID	Lists each unique field or column identified in your data source. Cells in this column cannot be edited.
Variable Name	Contains an automatically generated variable for each column in your data source. All generated variables are named Vn, where n is 0 for the first column and is incremented by 1 for each subsequent column in the source. To assign a different variable, click the appropriate cell and enter the new variable.
Variable Type	Contains a list for each column in your data source. Use the list to specify whether a variable is string or numeric.
Sample Value	Contains sample values from the first record of your source. These sample values help you identify the contents of each column of your source. Cells in the Sample Value column cannot be edited.
Contents	Contains a list for each column in your data source. Use the list to specify the type of value contained in each column of your source.
Formula	This column is grayed-out for all fields in your source, and becomes available only when you create a new variable. When you create a new variable, double-click the associated Formula cell to open the Process Variable Formula dialog box, from which you can define a formula for the variable.

Buttons

Button	Description
New variable	Click to create a new variable.
Delete	Click to delete a user-created variable.

Process Variable Formula

The Process Variable Formula dialog box displays and allows editing of formulas used in a TurboIntegrator process. When a formula exists, and you click Formula on the Variables tab, the currently set formula displays in the Formula window. Click New Variable to define a new formula.

Option	Description
Formula	The currently entered formula displays in this window. As you enter formula text this window updates.

Option	Description
Destination	Choose the location for this formula depending on your programming needs for this process. Select Data to put this formula into the Data section of the TurboIntegrator process. Select Metadata to position the formula in the MetaData section. Both puts the formula in both locations.
Evaluate	Click here to validate the formula.
Sample value	When the formula is evaluated, information about the formula displays here. For example, *Line 1: Syntax error on or before: \n (end of line) missing semicolon.*
Show automatically everytime the variable name changes	Click here to display this dialog box if the variable name is changed..If the box is unchecked, you must manually request it by clicking the Formula box on the Variables tab,

Maps Tab

Use the Maps tab to specify how source data maps to cubes, dimensions, data, consolidations, and attributes in the TM1 database.

The Maps tab consists of a series of sub-tabs, each containing options that let you map variables for your source data to existing TM1 metadata structures. The sub-tabs that are available vary according to the type of values contained in your source data, as specified in the Contents column of the Variables tab.

The Maps tab contains the following sub-tabs.

Cube

Use the Cube sub-tab to specify how TurboIntegrator maps imported data to TM1 cubes. The Cube sub-tab includes the following options.

Option	Description
Cube Action	Select an option to create, update, recreate, or apply no action to a cube.
Cube Name	Specify the cube to which the action applies. If creating a new cube, type the cube name in the entry field. Otherwise, select an existing cube from the list.
Zero Out Portion	This option becomes available when you select the Update Cube action. Select this box if you want to set all data points in a cube view to zero.

Option	Description
View Name	This option becomes available when you select the **Update Cube** and Zero Out Portion options. Select or define the view that encompasses the data points you want to zero out.
Data Action	Select an option that determines how processed data is stored in the cube. **Store Values** overwrites existing cube values with values imported by the process. **Accumulate Values** adds values imported by the process to existing cube values.
Enable Cube Logging	Fill this check box to write cube changes to the Tm1s.log file. Clear this box to process cubes without recording changes in Tm1s.log.

Dimensions

Use the Dimensions sub-tab to map element variables to dimension elements.

The sub-tab includes a grid you use to map individual variables to dimensions in the TM1 database. The grid includes the following columns.

Column	Description
Element Variable	Contains the name of each variable for which you specified a Contents value of Element. The Contents value is specified in the Variables tab. This column also contains the label (Data Variables) for any variables with a Contents value of Data.
Sample Value	A sample value from the first record of your data source. Use this value to help identify the dimension to which the element variable maps.
Dimension	Lists all dimensions available on the server. Select the dimension to which the element variable maps. To map the element variable to a new dimension, type the new dimension name in the entry field.

Column	Description
Order in Cube	This option becomes available when the Cube Action is Create. Specify the order of each dimension in the cube you are creating.
Action	Lists available dimension actions. Select an action. To create a new dimension, you must specify an action of **Create**.
Element Type	Select an element type for the variable, either **Numeric** or **String**.
Element Order	Select an option for ordering elements in any dimensions you are creating or updating. There are four sort orders: Input - Sorts elements in the order they are created in the dimension. Name - Sorts elements in alphabetical order, either ascending or descending. Level - Sorts elements by hierarchy level, either ascending or descending. Hierarchy - Sorts elements as they exist in the dimension hierarchy.

Data

Use the Data sub-tab to map data variables to specific elements.

The sub-tab includes a grid you use to map individual variables to elements in the TM1 database. The grid includes the following columns.

Column	Description
Data Variable	Contains the name of each variable for which you specified a Contents value of Data. The Contents value is specified in the Variables tab.
Element	Click the right arrow button to open the Subset Editor, where you can choose the element to which the variable maps. To map the variable to a new element, type the element name in the entry field.
Element Type	Select an element type here.

Column	Description
Sample Value	A sample value from the first record of your data source. Use this value to help identify the element to which the data variable maps.

Consolidations

Use the Consolidations sub-tab to map children to consolidated elements.

The sub-tab includes a grid you use to map individual variables to dimensions in the TM1 database. The grid includes the following columns.

Column	Description
Cons. Variable	Contains the name of each variable for which you specified a Contents value of Consolidation. The Contents value is specified in the Variables tab.
Dimension	List of dimensions to which the consolidation can map.
Child Variable	Lists the variables from which you select the immediate child of the consolidation.
Weight	Assigns a weight to the specified child variable.
Sample Value	A sample value from the first record of your data source. Use this value to help identify the element to which the consolidation maps.
Element Order	Select an option for ordering elements in any consolidations you are creating or updating. There are four sort orders: Input - Sorts elements in the order they are created in the dimension. Name - Sorts elements in alphabetical order, either ascending or descending. Level - Sorts elements by hierarchy level, either ascending or descending. Hierarchy - Sorts elements as they exist in the dimension hierarchy.

Attributes

Use the Attributes sub-tab to map attribute variables to specific attributes.

The sub-tab includes a grid you use to map individual variables to dimensions in the TM1 database. The grid includes the following columns.

Column	Description
Attribute Variable	Contains the name of each variable for which you specified a Contents value of Attribute. The Contents value is specified in the Variables tab.
Sample Value	Displays a sample value from the data source. Use this sample to help map the attribute.
Dimension	Lists all dimensions available on the server. Select the dimension to which the attribute applies.
Element Variable	Lists the element variables. Select the variable for the element to which the attribute variable applies.
Attribute	Lists the attributes to which the variable can map. Select the appropriate attribute from this list.
Action	Choose to either Create a new attribute or Update an existing one.
Attribute Type	Identifies the type of attribute selected in the Attribute column.

Advanced Tab

The Advanced tab contains several sub-tabs that display statements generated by TM1 based on the options you select elsewhere in the TurboIntegrator Editor. The Advanced tab also includes a sub-tab where you can define parameters for the process.

Parameters

Item	Description
Insert	Click to insert a new parameter.
Delete	Click to delete a selected parameter.
Parameters	Type a name for each new parameter.
Type	For each parameter, select a type here.
Default Value	Enter a value to use as the default value for this parameter when the TurboIntegrator process runs.

Item	Description
Prompt Question	Enter a prompt to use for this parameter when the TurboIntegrator process runs.

Prolog

Item	Description
Statement text box	Displays generated statements that define a series of actions to be executed before the data source is processed. You can enhance a process by creating additional statements with rules or TurboIntegrator functions.
Goto Line button	Click this button, enter the line you want to go to, then click OK to go directly to a line of code in the statement text box.

Metadata

Item	Description
Statement text box	Displays generated statements that define a series of actions to be executed on TM1 metadata before the data source is processed. You can enhance a process by creating additional statements with rules or TurboIntegrator functions.
Got Line button	Click this button, enter the line you want to go to, then click OK to go directly to a line of code in the statement text box.

Data

Item	Description
Statement text box	Displays generated statements that define a series of actions to be executed when the data source is processed. You can enhance a process by creating additional statements with rules or TurboIntegrator functions.

Item	Description
Goto Line button	Click this button, enter the line you want to go to, then click OK to go directly to a line of code in the statement text box.

Epilog

Item	Description
Statement text box	Displays generated statements that define a series of actions to be executed after the data source is processed. You can enhance a process by creating additional statements with rules or TurboIntegrator functions.
Goto Line button	Click this button, enter the line you want to go to, then click **OK** to go directly to a line of code in the statement text box.

Schedule Tab

Use this tab to schedule a process to execute at regular intervals.

Item	Description
Schedule this Process as a Chore Named	Check here to execute this process as a chore at regular intervals. By default, the chore bears the same name as the process. If you want to assign the chore a different name, type it in the entry field.
Chore Start Date and Time	Select a start date on the calendar and specify a start time in the Time field.
Chore Execution Frequency	Fill the appropriate fields to establish the interval at which the chore should be executed.

View Extract Window

Use the View Extract window to create a view that includes only those values satisfying user-defined criteria, or to define a view for export.

Skip parameters

Parameter	Description
Skip Consolidated Values	Turn this option on to ignore values derived through consolidation when extracting the view. Turn this option off to include values derived through consolidation when extracting the view. The default is off.

Parameter	Description
Skip Rule Calculated Values	Turn this option on to ignore values derived through rules when extracting the view. Turn this option off to include values derived through rules when extracting the view. The default is off.
Skip Zero/Blank Values	Turn this option on to ignore zeros or blank values when extracting the view. Turn this option off to include zeros or blank values when extracting the view. The default is on.

Range parameters

Parameter	Description
Operator	Select an operator that defines the values you want to extract.
Numeric Limits	Enter a numeric value for the variable(s) in the Operator.
Text Limits	Enter a string value for the variable(s) in the Operator.

Dimension Elements selection

For each dimension, click the **Subset** button and select the elements or subset that defines the parameters for the view extract.

If the view from which you are creating the extract contains more than 16 dimensions, click ◀ to page backward to the previous 16 dimensions, or click ▶ to page forward to the next 16 dimensions.

View Styles Dialog Box

The View Styles dialog box lets you apply Excel styles to the TM1 cube view in the In-Spreadsheet Browser. The dialog box contains several lists that let you apply an existing Excel style to a range of cells, as well as buttons that let you edit or create styles.

Item	Description
Background	Select a style from this list to apply to the background of the In-Spreadsheet Browser.
Data Cells	Select a style from this list to apply to data cells. The Data Cells style takes precedence over the Background style.

Item	Description
Row Header Cells	Select a style from this list to apply to row header cells. The Row Header Cells style takes precedence over the Background style.
Column Header Cells	Select a style from this list to apply to column header cells. The Column Header Cells style takes precedence over the Background style.
Edit Style buttons	Click the appropriate Edit Style button to edit or create styles for the associated range of the In-Spreadsheet Browser.
Freeze Panes	Toggle this option to freeze and unfreeze panes in the In-Spreadsheet Browser. When this option is toggled on, row element names remain visible when you scroll horizontally through a view, and column element names remain visible when you scroll vertically. When this option is toggled off, row and column element names move along with cube values as you scroll through a view.

Chapter 3. Rules Functions

This section contains a complete list of all IBM Cognos TM1 rules functions. You can use any of these functions when writing TM1 rules.

You can also incorporate all rules functions, with the exception of the STET and ISLEAF functions, in TurboIntegrator processes.

Arithmetic Operators in TM1 Rules

The following mathematical operators can be used when constructing TM1 rules.

Operator	Meaning
+ (plus sign)	Addition
- (minus sign)	Subtraction
* (asterisk)	Multiplication
/ (forward slash)	DivisionDivision by zero using this operator returns an undefined value.
\ (back slash)	DivisionDivision by zero using this operator returns zero.
^ (caret/circumflex)	Exponentiation

Comparison Operators in TM1 Rules

The comparison operators compare values in the formula portion of a rule calculation statement.

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 in the following example:

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

Logical Operators in TM1 Rules

You can combine expressions in a rules calculation statement using logical operators.

Operator	Meaning	Example
& (ampersand)	AND	(Value1 > 5) & (Value1 < 10) Returns TRUE if the value is greater than 5 and less than 10.
% (percentage sign)	OR	(Value1 > 10) % (Value1 < 5) Returns TRUE if the value is greater than 10 or less than 5.
~ (tilde)	NOT	~(Value1 > 5) Equivalent to (Value1 <= 5)

Consolidation Calculation Rules Functions

The ConsolidatedMax; ConsolidatedMin; ConsolidatedAvg; ConsolidatedCount; and Consolidated CountUnique perform mathematical calculations on consolidations.

ConsolidatedMin

ConsolidatedMin calculates the minimum value in a consolidation and returns a single value.

This is a TM1 Rules function, valid in Rules and TurboIntegrator processes.

Syntax

```
ConsolidatedMin(flag-value, cube-name, element_1, element_2,... );
```

Arguments

flag-value

The flag value is the sum of the following values:

1 - use consolidation weighting when computing the value. If this is not turned on the raw value of the consolidated element will be used. ⌂

2 - ignore zero values. If this is set, zero values will not be used by these functions.

Note: If zero is specified as the weighting of some consolidated elements, then the configuration parameter: ZeroWeightOptimization=F must be set for these elements to be factored into the computation of these functions. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not processed when discovering things line the minimum or maximum value.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a Turbo Integrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedMin( 1, '', !actvsbud, '1 Quarter' );`

`element_1, element_2, ...`

Dimension element names that define the intersection of the cube containing the value to be retrieved.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

ConsolidatedMax

ConsolidatedMax calculates the maximum value in a consolidation and returns a single value.

This is a TM1 Rules function, valid in Rules and TurboIntegrator processes.

Syntax

```
ConsolidatedMax(flag-value, cube-name, element_1, element_2,... );
```

Arguments

`flag-value`

The flag value is the sum of the following values:

1 - use consolidation weighting when computing the value. If this is not turned on the raw value of the consolidated element will be used. ⌂

2 - ignore zero values. If this is set, zero values will not be used by these functions.

Note: If zero is specified as the weighting of some consolidated elements, then the configuration parameter: ZeroWeightOptimization=F must be set for these elements to be factored into the computation of these functions. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not processed when discovering things line the minimum or maximum value.

`cube-name`

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a Turbo Integrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedMax( 1, '', !actvsbud, '1 Quarter' );`

`element_1, element_2, ...`

Dimension element names that define the intersection of the cube containing the value to be retrieved.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

Consider a cube called Income Statement with three dimensions, "Area", "Time", and "Income Statement". The Income Statement dimension contains an element "Gross Sales" for the overall sales number.

To calculate the maximum sales across all regions in the year 2010 use:

```
ConsolidatedMax( 1,  'Income Statement', 'All Regions', '2010', 'Gross Sales' );
```

ConsolidatedAvg

ConsolidatedAvg calculates the average value in a consolidation and returns a single value.

This is a TM1 Rules function, valid in Rules and TurboIntegrator processes.

Syntax

```
ConsolidatedAvg(flag-value, cube-name, element_1, element_2,... );
```

Arguments

`flag-value`

The flag value is the sum of the following values:

1 - use consolidation weighting when computing the value. If this is not turned on the raw value of the consolidated element will be used. ⌂

2 - ignore zero values. If this is set, zero values will not be used by these functions.

Note: If zero is specified as the weighting of some consolidated elements, then the configuration parameter: ZeroWeightOptimization=F must be set for these elements to be factored into the computation of these functions. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not processed when discovering things line the minimum or maximum value.

`cube-name`

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a Turbo Integrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedAvg( 0, '', !actvsbud, '1 Quarter' );`

`element_1, element_2, ...`

Dimension element names that define the intersection of the cube containing the value to be retrieved.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

In a cube called Income Statement with three dimensions: Regions, Time, and Income Statement, the Income Statement dimension contains an element called Gross Sales for the overall sales number.

To calculate the average sales across all regions in the year 2010 write:

```
ConsolidatedAvg( 1,  'Income Statement', 'All Regions', '2010', 'Gross Sales' );
```

ConsolidatedCount

ConsolidatedCount counts the value in a consolidation and returns a single value.

This is a TM1 Rules function, valid in Rules and TurboIntegrator processes.

Syntax

```
ConsolidatedCount(flag-value, cube-name, element_1, element_2,... );
```

Arguments

`flag-value`

The flag value is the sum of the following values:

1 - use consolidation weighting when computing the value. If this is not turned on the raw value of the consolidated element will be used. ⌂

2 - ignore zero values. If this is set, zero values will not be used by these functions.

Note: If zero is specified as the weighting of some consolidated elements, then the configuration parameter: ZeroWeightOptimization=F must be set for these elements to be factored into the computation of these functions. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not processed when discovering things line the minimum or maximum value.

`cube-name`

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a Turbo Integrator process, the cube-name argument can be specified as an empty string to mean the current cube. This means you may write a rule such as:`['Apr']=ConsolidatedCount( 1, '', !actvsbud, '1 Quarter' );`

`element_1, element_2, ...`

Dimension element names that define the intersection of the cube containing the value to be retrieved.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

ConsolidatedCountUnique

ConsolidatedCountUnique counts the number of unique elements for which data points actually exist for the specified consolidation.

This is a TM1 Rules function, valid in Rules and TurboIntegrator processes.

The unique elements are counted along one dimension of the consolidated cell.

Syntax

```
ConsolidatedCountUnique( 0, unique-along-dimension-name, cube-name,elem_1, elem_2, . . . );
```

Arguments

flag word
Currently specify zero

Note: If zero is specified as the weighting of some consolidated elements, then the configuration parameter: ZeroWeightOptimization=F must be set for these elements to be factored into the computation of these functions. Without this configuration parameter, the elements for which the weighting is zero are eliminated from the consolidation list, and are therefore not processed when discovering things line the minimum or maximum value.

unique-along-dimension-name
The dimension along which unique element entries for which there is real data are to be counted.

cube-name

Name of the cube where the values reside.

If the function is running as part of a cube rule, and NOT as part of a Turbo Integrator process, the cube-name argument can be specified as an empty string to mean the current cube.

element_1, element_2, ...

Dimension element names that define the intersection of the cube which is the consolidated value to be processed.

Arguments element_1 through element_n are sequence-sensitive. element_1 must be an element from the first dimension of the cube, element_2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

In a cube called Income Statement with three dimensions: Regions, Time, and Income Statement, the Income Statement dimension contains an element called Gross Sales for the overall sales number. To count how many regions had some gross sales in the year 2010 write:

```
ConsolidatedCountUnique( 0, 'Regions', 'Income Statement',
'All Regions', '2010', 'Gross Sales' );
```

Cube Data Rules Functions

Rules functions that work on cube data.

DB

DB returns a value from a cube in a TM1 database. DB returns a numeric value if used in a numeric expression and a string value if used in a string expression.

This is a TM1 rules function, valid in TM1 rules only. Use of this function in a TurboIntegrator process will cause an error.

Syntax

```
DB(cube, e1, e2, [...e256])
```

Parameters

cube

The name of the cube from which to retrieve the value.

e1,...en

Dimension element names that define the intersection containing the value to be retrieved.

Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on.

Example

```
DB('Budget', 'California', '15" Flat Panel Monitors', 'Net Sales', 'January')
```

In this example, Budget is the cube name, and the function returns the value at the intersection of California, 15" Flat Panel Monitors, Net Sales, and January.

ISLEAF

ISLEAF returns 1 if a specified cell is a leaf cell (identified solely by leaf/simple elements). If the specified cell is identified by any consolidated elements, the function returns 0.

This is a TM1 rules function, valid only in TM1 rules.

The ISLEAF function cannot be used in TurboIntegrator processes. The presence of this function in a process will prevent the process from compiling.

Syntax

```
ISLEAF
```

Arguments

None.

Example

You can use ISLEAF in an IF statement to test if a current cell is a leaf cell. For example,

```
[]=IF((ISLEAF=1),TrueStatement, FalseStatement);
```

Executes the TrueStatement if the current cell is a leaf cell, otherwise it executes the FalseStatement.

ISUNDEFINEDCELLVALUE

ISUNDEFINEDCELLVALUE compares the passed value to the default numeric cube value, which is influenced by the presence of the UNDEFVALS declaration in that cube's rule. The function returns 1 if the passed value equals the cube's default value, otherwise the function returns 0.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ISUNDEFINEDCELLVALUE(TestValue, <Cube>)
```

Arguments

Argument	Description
TestValue	The numerical value to compare against the cube's default value.
Cube	An optional String argument that specifies the cube whose default value should be compared. When ISUNDEFINEDCELLVALUE is used in a rule, the cube is assumed to be the subject cube unless otherwise specified. When used in a TI, a cube should be specified. If the cube is omitted in a TI, or is not valid when specified, 0 will be used for comparison.

Example

`ISUNDEFINEDCELLVALUE(TestValue)` returns 1 when TestValue is the special undefined value and is used in the rule of a cube with UNDEFVALS declared.

UNDEF

UNDEF returns the undefined value. This function can be used to prevent datafrom being stored in a cube based on a logical test.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
UNDEF
```

Arguments

None.

Example

UNDEF returns the undefined value.

UNDEFINEDCELLVALUE

UNDEFINEDCELLVALUE returns the default numeric cube value, which is influenced by the presence of the UNDEFVALS declaration in that cube's rule.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
UNDEFINEDCELLVALUE(<Cube>)
```

Arguments

Argument	Description
Cube	An optional String argument that specifies the cube whose default value should be returned. When UNDEFINEDCELLVALUE is used in a rule, the cube is assumed to be the subject cube unless otherwise specified. When used in a TI process, a cube should be specified. If the cube is omitted in a TI process, or is not valid when specified, 0 will be returned.

Example

UNDEFINEDCELLVALUE returns 0 when used in the rule of a cube without UNDEFVALS declared, or when used in a TI process.

UNDEFINEDCELLVALUE returns the special undefined value when used in the rule of a cube with UNDEFVALS declared.

`UNDEFINEDCELLVALUE('ExampleCube')` returns the default value of 'ExampleCube', or 0 if 'ExampleCube' does not exist.

UNDEFVALS

Putting UNDEFVALS in the rules for a cube changes the default value for the cube from zero to an *special undefined* value.

This is a TM1 rules function, valid in both TM1 rules and TM1 TurboIntegrator processes.

This has ramifications for how data is stored in the cube and retrieved.

- Data Storage

  For a cube without UNDEFVALS in the rules, the default value is zero. If an attempt is made to store a zero in a cell of the cube, that storage request is ignored, since one is trying to store the default value, and it would needlessly consume memory space.

  If however the cube has UNDEFVALS defined in the rules, this makes the default value a *special undefined* value. Now when a zero is stored in a cell of a cube, it is actually stored, just like any other non-zero value.

- Data Retrieval

  For a cube without UNDEFVALS in the rules, the default value is zero. When a cell is retrieved, and there is no value currently stored for that value in the cube, a value of zero (as the default value) is returned.

  If however the cube has UNDEFVALS defined in the rules, this make the default value a *special undefined* value. Now when a non-existent cell is retrieved, the value retrieved will be this *special undefined* value. This can be used to distinguish a cell which does not exist (*undefined value*) from a cell which exists but whose value is zero (zero returned).

Note: It is not currently possible in Turbo Integrator to test for this special undefined value directly. This special undefined value is not the value returned by the UNDEF() Turbo Integrator function. The value returned by UNDEF() is an undefined value used for such things as an attempt to divide by zero, or take the logarithm of an illegal number, etc.

You can indirectly test for this special undefined value as follows:

```
vv = CellGetN( 'cube-name', elements-list);if ( vv >
0 );
   # 'special undefined' value is greater than zero
   # convert the value to a string
   vv_str = str(vv,20,16);
   # convert the string back to a number
   converted_vv = NUMBR( vv_str );
   if ( converted_vv = 0 );
      #converted number is now zero, but original was non-zero
      #this is the 'special undefined' value
   Else
      #normal non-zero cell value
   Endif;
Endif;
```

Syntax

```
UNDEFVALS
```

Arguments

None.

Date and Time Rules Functions

Rules functions that work with dates and time.

DATE

DATE returns the date string in *yy-mm-dd* or *yyyy-mm-dd* format for a given serial number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

DATE(*SerialNumber*, *ReturnFourDigitYear*)

Argument	Description
SerialNumber	A date expressed in serial format.

Argument	Description
ReturnFourDigitYear	An optional Boolean argument that determines whether the DATE function returns a string using two- or four-digit notation for the year. If ReturnFourDigitYear is true, the function returns date falling within the range of Jan. 1, 1960 and Dec. 31, 9999, using four-digit notation for the year. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ReturnFourDigitYear is false, or if this optional argument is omitted from the DATE function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059, using two-digit notation for the year. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ReturnFourDigitYear is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, DATE uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

DATE(13947) returns `98-03-09`.

DATE(13947, 1) returns `1998-03-09`.

DATES

DATES returns a date string, in the form 'yy-mm-dd' or 'yyyy-mm-dd', corresponding to a given year, month, and day.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DATES(year, month, day)
```

Argument	Description
year	A year, expressed in either yy or yyyy format.
month	A month, expressed in mm format.
day	A day, expressed in dd format.

Example

DATES(98, 2, 10) returns '98-02-10'.

DATES(1998, 2, 10) returns '1998-02-10'.

DAY

DAY returns a numeric value for the day in a given date string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DAY(DateString)
```

Argument	Description
DateString	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

DAY('02-05-25') returns 25.

DAYNO

DAYNO returns the serial date number corresponding to a given date string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Note: DAYNO can return serial dates for date strings starting at January 1, 1960 (dates string 1960-01-01 or 60-01-01). For dates after December 31, 2059, you use a four digit year in the date string. For example, the date string for January 5, 2061 would be 2061-01-05.

Syntax

```
DAYNO('DateString')
```

Argument	Description
DateString	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

DAYNO('98-03-09') returns 13947.

MONTH

MONTH returns a numeric value for the month in a given date string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MONTH(date)
```

Argument	Description
date	A date string in either YY-MM-DD or YYYY-MM-DD format.

Example

MONTH('02-05-25') returns 5.

NOW

NOW returns the current date/time value in serial number format.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
NOW
```

Arguments

None.

Example

NOW returns the current date/time value in serial number format.

TIME

TIME returns a string, in HH:MM format, representing the system time on the TM1 server.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIME
```

Arguments

None.

Example

Given a system time of 9:33 AM, TIME returns the string '09:33'.

Given a system time of 9:33 PM, TIME returns the string '21:33'.

TIMST

TIMST returns a formatted date/time string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIMST(datetime, format, ExtendedYears)
```

Argument	Modifier/ Description
datetime	A date/time serial number. The integer part of the number specifies the date, and the decimal part specifies the time within the day. Day number 0 corresponds to '60-01-01'. Negative numbers correspond to prior years. Years in the 21st Century, up to 2059, are represented by years 00 through 59. An hour is 1/24th of a day, a minute 1/60th of an hour, and a second 1/60th of a minute.
format	A string that formats the result of the function. All the characters in the format argument appear in the result, except for the following characters, which return date/time component values:
	\y the last two digits of the year (97, 98, etc.)
	\Y the four digits of the year (1997, 1998, etc.)
	\m the two digits of the month (01 through 12)
	\M the abbreviation of the month (JAN, FEB, etc.)
	\d the two digits of the day (01 through 31)
	\D the digit of the day (1 through 31)

Argument	Modifier/ Description
	\h the hour in military time (00 through 23)
	\H the standard hour (1 through 12)
	\i the minute (00 through 59)
	\s the second (00 through 59)
	\p a.m. or p.m.
ExtendedYears	This optional Boolean parameter specifies whether the function returns a date falling within the range 1960 - 2059 or 1960 - 9999. If ExtendedYears is true, the function returns a date falling within the range of Jan. 1, 1960 and Dec. 31, 9999. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ExtendedYears is false, or if this optional argument is omitted from the TIMST function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ExtendedYears is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TIMST uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

TIMST(366.0000, '\M \D, \Y') returns 'JAN 1, 1961'.

TIMST(366.5000, '\H\p \imin\ssec') returns '12p.m. 00min00sec'.

TIMST(366.1000, 'On \M \D, \Y at \H\p \imin\ssec') returns 'On JAN 1, 1961 at 2a.m. 24min00sec'.

TIMST(11111.1100, 'On \M \D, \Y at \H\p \imin\ssec') returns 'On JUN 3,1990 at 2a.m. 38min24sec'.

TIMVL

TIMVL returns the numeric value of a component (year, month, etc.) of a date/time value.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TIMVL(datetime, type, ExtendedYears)
```

Argument	**Modifier/** **Description**
datetime	A date/time serial number. The integer part of the number specifies the date, and the decimal part specifies the time within the day. Day number 0 corresponds to '60-01-01.' Negative numbers correspond to prior years. Years in the 21st Century, up to 2059, are represented by years 00 through 59. An hour is 1/24th of a day, a minute 1/60th of an hour, and a second 1/60th of a minute.
type	A character that specifies the type of component to be extracted. The following are valid type arguments:
	Y year value (1997, 1998, etc.)
	M month value (1 through 12)
	D day value (1 through 31)
	H hour value (0 through 23)
	I minute value (00 through 59)

Argument	Modifier/ Description
	S second value (00 through 59)
ExtendedYears	This optional Boolean parameter specifies whether the function returns a date falling within the range 1960 - 2059 or 1960 - 9999. If ExtendedYears is true, the function returns a date falling within the range of Jan. 1, 1960 and Dec. 31, 9999. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ExtendedYears is false, or if this optional argument is omitted from the TIMVL function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ExtendedYears is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TIMVL uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

TIMVL(11111.1100, 'Y') returns 1990.

TIMVL(11111.1100, 'H') returns 2.

TODAY

TODAY returns the current date in yy-mm-dd format.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TODAY(<ReturnFourDigitYear>)
```

Argument	Description
ReturnFourDigitYear	An optional Boolean argument that determines whether the TODAY function returns a string using two- or four-digit notation for the year. If ReturnFourDigitYear is true, the function returns date falling within the range of Jan. 1, 1960 and Dec. 31, 9999, using four-digit notation for the year. Serial date 0 corresponds to Jan. 1, 1960 and serial date 2936549 corresponds to Dec. 31, 9999. If ReturnFourDigitYear is false, or if this optional argument is omitted from the TODAY function, the function returns a date falling within the range Jan. 1, 1960 and Dec. 31, 2059, using two-digit notation for the year. Serial date 0 corresponds to Jan 1, 1960 and serial date 36524 corresponds to Dec. 31, 2059. If ReturnFourDigitYear is false or is omitted and you specify a serial date greater than 36524, the serial date used by the function is determined by the formula n - 36525. For example, if you specify a serial date of 36530, then 36530 - 36525 = 5. In this case, TODAY uses 5 as the serial date and returns the date Jan. 6, 1960.

Example

P1=TODAY(1) returns a data string in YYYY-MM-DD format such as 2009-06-05.

P1=TODAY(0) returns a date string in YY-MM-DD format such as 09-06-05

YEAR

YEAR returns a numeric value for the year in a given date string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
YEAR(date)
```

Argument	Description
date	A date string in YY-MM-DD format.

Example

YEAR('02-05-25') returns 2.

Dimension Information Rules Functions

Rules functions that manage dimension information.

ATTRN

ATTRN returns a numeric attribute for a specified element of a dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATTRN(dimension, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
element	An element of the dimension.

attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.
	Note: : When this function is used in a conditional statement (IF), the statement is the portion containing the condition, not the entire conditional block. After a minor error, execution continues with the next statement. TI processing has no knowledge that it was in a conditional once the minor error is processed, so the next statement is the next line, not the line after the endif.
	To avoid this situation, use variables for any operation that could encounter a minor error and then use the variables in the conditional statement. For example:
	`V1 = CELLGETN('PNLCube', 'fred', 'argentina','Sales','Jan');`
	`IF(V1 = 454);ASCIIOUTPUT ('bug.txt', 'if logic not working properly');`
	`ENDIF;`

Example

ATTRN('Model', 'L Series 1.8L Sedan', 'Engine Size')

In this example, the function returns the numeric value of the Engine Size attribute of the L Series 1.8L Sedan element in the Model dimension.

ATTRS

ATTRS returns a string attribute for a specified element of a dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATTRS(dimension, element, attribute)
```

Argument	Description
dimension	A valid dimension name.
element	An element of the dimension.
attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Example

ATTRS('plan_business_unit', '10100', 'Currency')

In this example, the function returns the string value of the Currency attribute of the 10100 element in the plan_business_unit dimension.

ConsolidateChildren

ConsolidateChildren forces consolidated values to be calculated by summing immediate children along a specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

ConsolidateChildren is useful when intermediate consolidations are calculated by rules and you want a parent consolidation to be calculated by summing the intermediate consolidations rather than by summing the underlying leaf values.

Syntax

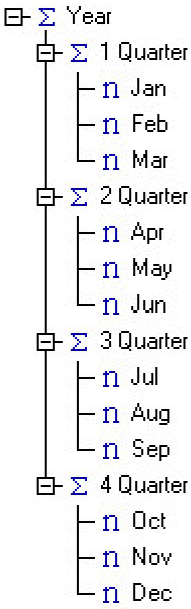
```
ConsolidateChildren(DimName1, DimName2, ...)
```

Argument	Description
DimName1, DimName2, ...	Names of the dimensions along which consolidations will be performed. The function requires at least one DimName argument, and can accept as many DimName arguments as there are dimensions in the cube for which the rule is written.

Example

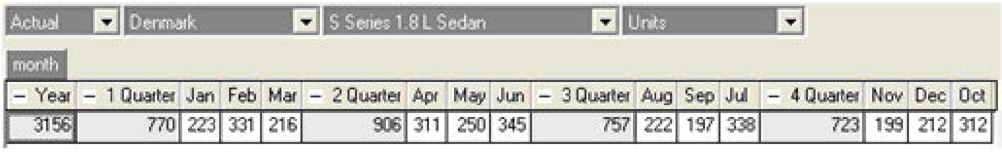
Consider a cube named Sales composed of the dimensions ActVsBud, Region, Model, Account1, and Month.

In this example, the Month dimension is defined as follows:



- Year
  - 1 Quarter
    - Jan
    - Feb
    - Mar
  - 2 Quarter
    - Apr
    - May
    - Jun
  - 3 Quarter
    - Jul
    - Aug
    - Sep
  - 4 Quarter
    - Oct
    - Nov
    - Dec

If no rule is in place for this cube, the value of the Year consolidation is calculated by summing all the underlying leaf values, in this case Jan through Dec. The following figure illustrates this consolidation.



Actual | Denmark | S Series 1.8 L Sedan | Units

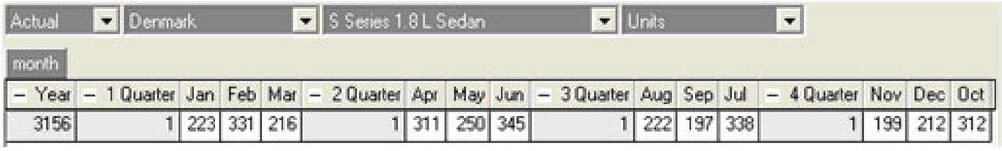
month

− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
3156	770	223	331	216	906	311	250	345	757	222	197	338	723	199	212	312

Now, suppose you create the following rule for this cube, which indicates that all quarterly values should be 1:

```
[{'1 Quarter', '2 Quarter', '3 Quarter', '4 Quarter'}]=1;
```

The result is as follows:



Actual | Denmark | S Series 1.8 L Sedan | Units

month

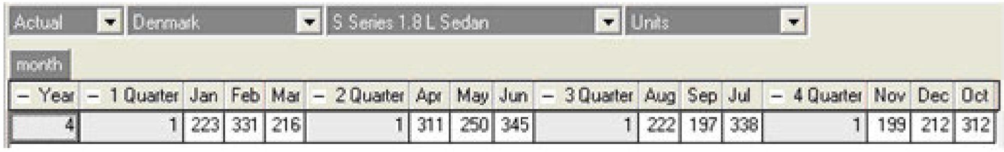
− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
3156	1	223	331	216	1	311	250	345	1	222	197	338	1	199	212	312

In the figure, you can see that quarterly values are indeed calculated by the rule, but the Year consolidation is still calculated by summing all underlying leaf values. If this is not your desired calculation path, you can use the ConsolidateChildren function to force TM1 to calculate the Year consolidation by summing its immediate children, specifically 1 Quarter, 2 Quarter, 3 Quarter, and 4 Quarter.

```
['Year']=ConsolidateChildren('Month');[{'1 Quarter', '2 Quarter', '3 Quarter', '4 Quarter'}]=1;
```

In the rule, the statement `['Year']=ConsolidateChildren('Month')` says that the Year consolidation should be calculated by summing the immediate children of Year in the Month dimension.

The following figure shows the result of the `['Year']=ConsolidateChildren('Month')` statement:



Actual | Denmark | S Series 1.8 L Sedan | Units

month

− Year	− 1 Quarter	Jan	Feb	Mar	− 2 Quarter	Apr	May	Jun	− 3 Quarter	Aug	Sep	Jul	− 4 Quarter	Nov	Dec	Oct
4	1	223	331	216	1	311	250	345	1	222	197	338	1	199	212	312

Note that the Year consolidation is now calculated by summing its immediate children.

It's important to remember that for a given consolidation, the ConsolidateChildren function applies only to the *immediate* children of the consolidation.

The ConsolidateChildren function can also be used to specify how consolidations are calculated in multiple dimensions, as in the following example:

Argument	Description
['World','Year']= ConsolidateChildren('Region','Month')	This statement applies the ConsolidateChildren function to both the World and Year consolidations. In this case, World is calculated by summing all its immediate children in the Region dimension, while Year is calculated by summing its immediate children in the Month dimension.

DIMNM

DIMNM returns the element of a dimension that corresponds to the index argument.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMNM(server_name:dimension, index)
```

Argument	Description
dimension	A valid dimension name qualified by the server name.

Argument	Description
index	A value less than or equal to the number of elements in the dimension. If this argument is less than 1, or greater than the number of elements in the dimension, the function returns 0.

Example

DIMNM(planning_sample:'Region',2)

This example returns 'Belgium', which is the element within the Region dimension with an index value of 2.

DIMSIZ

DIMSIZ returns the number of elements within a specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMSIZ(dimension)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.

Example

DIMSIZ('Accounts')

If the dimension Accounts contains 19 elements, the example returns the value 19.

DNEXT

DNEXT returns the element name that follows the element specified as an argument to the function.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DNEXT(dimension, element)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

DNEXT("Location","Oregon")

If the Location dimension contains the ordered elements California, Oregon, and Washington, the example returns Washington.

DNLEV

DNLEV returns the number levels in a dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

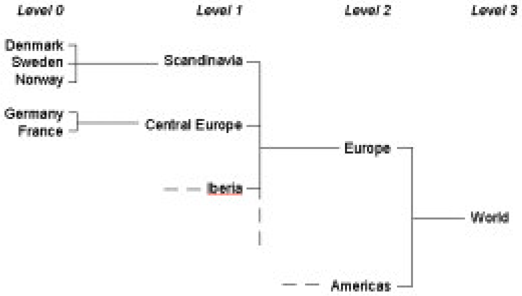
```
DNLEV(dimension)
```

Argument	Description
dimension	A valid dimension name. Some installations may need to qualify the dimension name with the server name, as in `server_name:dimension`.

Example

```
DNLEV('Region')
```

In the Region dimension, the various nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3).



There are four levels in the Region dimension, so the example returns the value 4.

TABDIM

TABDIM returns the dimension name that corresponds to the index argument.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TABDIM(cube, index)
```

Argument	Description
cube	A valid cube name.
index	A positive value less than or equal to the total number of dimensions in the cube.

Example

TABDIM('SalesCube',3)

The cube SalesCube contains five dimensions: account1, actvsbud, model, month, and region. The example returns model, the third dimension of SalesCube.

Element Information Rules Functions

Rules functions that manage element information.

DIMIX

DIMIX returns the index number of an element within a dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DIMIX(server_name:dimension, element)
```

Argument	Description
dimension	A valid dimension name qualified by the server name.
element	The name of an element within the dimension. If the element is not a member of the dimension specified, the function returns 0.

Example

DIMIX(planning_sample:'Region','Brazil')

Brazil has an index value of three in the Region dimension. The example returns 3.

DTYPE

DTYPE returns information about the element type of a specified element.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

DTYPE returns N if the element is a numeric element, S if the element is a string element, and C if the element is a consolidated element.

Syntax

```
DTYPE(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

DTYPE('Region','Europe')

The element Europe is a consolidated element of the Region dimension, so the example returns C.

ELCOMP

ELCOMP returns the name of a child of a consolidated element in a specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

If the element argument is not a consolidated element, the function returns 0.

Syntax

```
ELCOMP(dimension, element, position)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension.
position	A positive value less than or equal to the total number of children in the specified element.

Example

ELCOMP('Region','Central Europe',2)

In the dimension Region, the consolidated element Central Europe is a consolidation of the children France and Germany. Germany is in the second position in this consolidation. Accordingly, the example returns Germany.

ELCOMPN

ELCOMPN returns the number of components in a specified element.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

If the element argument is not a consolidated element, the function returns 0.

Syntax

```
ELCOMPN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension.

Example

ELCOMPN('Region','Scandanavia')

In the Region dimension, the element Scandanavia is a consolidation of three elements. The example returns 3.

ELISANC

ELISANC determines whether element1 is an ancestor of element2 in the specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The function returns 1 if element1 is an ancestor of element2, otherwise the function returns 0.

Syntax

```
ELISANC(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

ELISANC('Region', 'Europe', 'Germany')

In the dimension Region, the element Europe is an ancestor of Germany. The example returns 1.

ELISCOMP

ELISCOMP determines whether element1 is a child of element2 in the specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The function returns 1 if element1 is a child of element2, otherwise the function returns 0.

Syntax

```
ELISCOMP(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

ELISCOMP('Region','Germany','Central Europe')

In the dimension Region, the element Central Europe is a consolidation of two elements, Germany and France. The example returns 1.

Note: this function returns 1 only for immediate children. In the above example, Germany is a child of Central Europe. Further, Central Europe is a child of Europe. However, because the function returns 1 only for immediate children, the following example returns 0:

ELISCOMP('Region','Germany','Europe')

ELISPAR

ELISPAR determines whether element1 is a parent of element2 in the specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The function returns 1 if element1 is a parent of element2, otherwise the function returns 0.

Syntax

```
ELISPAR(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension.
element2	The name of an element within the dimension.

Example

ELISPAR('Region','Central Europe','Germany')

In the dimension Region, the consolidated element Central Europe is the parent of both Germany and France. Accordingly, the example returns 1.

Note: this function returns 1 only for immediate parents. In the above example, Europe is a parent of Central Europe. Further, Central Europe is a parent of Germany. However, because Europe is not an immediate parent of Germany, the following example returns 0:

Note: ELISPAR('Region','Europe','Germany')

ELLEV

ELLEV returns the level of an element within a dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

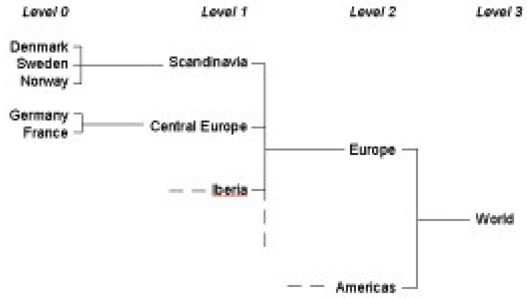
```
ELLEV(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

```
ELLEV('Region','Europe')
```

In the Region dimension, individual nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3). The example returns 2, as Europe is a Level 2 element.



ELPAR

ELPAR returns the parent of an element in a specified dimension

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELPAR(dimension, element, index)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.
index	A positive value less than or equal to the total number of consolidated elements (parents) that use the element argument as a child.

Example

ELPAR('Model','Wagon 4WD',2)

In the dimension Model, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. In the structure of the Model dimension, Total Wagons is defined first, Total 4WD is defined second.

The example returns Total 4WD, as this is the second instance of a parent to Wagon 4WD within the Model dimension.

ELPARN

ELPARN returns the number of parents of an element in a specified dimension.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELPARN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension.

Example

ELPARN('Model','Wagon 4WD')

In the Model dimension, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. The function returns 2.

ELWEIGHT

ELWEIGHT returns the weight of a child in a consolidated element.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ELWEIGHT(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of a consolidated element within the dimension.
element2	The name of a child of the consolidated element.

Example

ELWEIGHT('Account1','Gross margin','Variable Costs')

The element Variable Costs, which is a child of Gross margin, has a weight of -1.

The example returns -1.

Financial Rules Functions

Rules functions used to manage financial information.

FV

FV returns the value of an annuity at the time of the last payment.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

An annuity is a series of payments made at equal intervals of time. Payments are assumed to be made at the end of each period.

Syntax

```
FV(payment, interest, periods)
```

Argument	Description
payment	The amount of the payment made per period.
interest	The interest rate paid per period.
periods	The number of periods in the annuity.

Example

FV(1000, .14, 5)

This example returns the value of an annuity at the end of 5 years, with payments of $1,000 per year at 14% interest.

PAYMT

PAYMT returns the payment amount of an annuity based on a given initial value or principal, an interest rate, and a number of periods. An annuity is a series of payments made at equal intervals of time.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
PAYMT(principal, interest, periods)
```

Argument	Description
principal	The present value, or the total amount that a series of future payments is worth now.
interest	The interest rate paid per period.
periods	The number of periods in the annuity. Payments are assumed to be made at the end of each period.

Example

PAYMT(100000, .14, 5)

This example returns the payment on a 5-year annuity that is paid yearly, with a principal of $100,000 at 14% interest.

PV

PV returns the initial or principal value of an annuity.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
PV(payment, interest, periods)
```

Argument	Description
payment	The amount of the payment made.
interest	The interest rate paid per period.
periods	The number of periods in the annuity. Payments are assumed to be made at the end of each period.

Example

PV(1000, .14, 5)

This example returns the principal value of an annuity with 5 yearly payments of $1,000 at 14% interest.

Logical Rules Functions

Logical operators to use in rules.

CONTINUE

When included as part of a rules expression, CONTINUE allows a subsequent rule with the same area definition to be executed.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Normally, TM1 only executes the first rule encountered for a given area.

Syntax

```
CONTINUE
```

Arguments

None.

Example

['Jan']= if(!region @= 'Argentina',10,CONTINUE);

['Jan']=20;

In this example, all cells identified by January and Argentina are assigned a value of 10. Cells identified by Jan and any other Region element are assigned a value of 20.

IF

IF returns one value if a logical expression you specify is TRUE and another value if it is FALSE.

This is a TM1 rules function, valid only in TM1 rules. (TurboIntegrator uses its own IF function that is capable of evaluating multiple logical expressions.)

Syntax

```
IF(expression, true_value, false_value)
```

Argument	Description
expression	Any value or expression that can be evaluated to TRUE or FALSE.
true_value	The value that is returned if expression is TRUE.
false_value	The value that is returned if expression is FALSE.

Example

IF(1<2, 4, 5) returns 4.

IF(1>2, 'ABC', 'DEF') returns 'DEF'.

STET

The STET function cancels the effect of a rule for a particular element.

This is a TM1 rules function, valid only in TM1 rules. This function cannot be used in TurboIntegrator processes.

Syntax

```
STET
```

Arguments

None.

Example

['Sales'] = IF(!Region @= 'France',STET, 100);

In this example, the rule dictates that the value for Sales is always 100, except for the intersection of Sales and the element France from the Region dimension.

Mathematical Rules Functions

Mathematical operators to use in rules.

ABS

ABS returns the absolute value of a number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ABS(x)
```

Argument	Description
x	The number for which you want to find the absolute value.

Example

ABS(-1.2) returns 1.2

ACOS

ACOS returns the angle, in radians, whose cosine is x.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ACOS(x)
```

Argument	Description
x	The cosine of the angle you want to find. x must be between -1 and 1; otherwise the function returns an error.

Example

ACOS(0) returns 1.5708.

ASIN

ASIN returns the angle, in radians, whose sine is x.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ASIN(x)
```

Argument	Description
x	The sine of the angle you want to find. x must be between -1 and 1; otherwise the function returns an error.

Example

ASIN(1) returns 1.5708.

ATAN

ATAN returns the angle, in radians, whose tangent is x. The result is between -pi/2 and +pi/2.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ATAN(x)
```

Argument	Description
x	The tangent of the angle you want to find.

Example

ATAN(1) returns .7854.

COS

COS returns the cosine of an angle expressed in radians.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
COS(x)
```

Argument	Description
x	An angle, expressed in radians, for which you want to find the cosine.

Example

COS(0) returns 1.

EXP

EXP returns the natural anti-log of a number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
EXP(x)
```

Argument	Description
x	A number for which you want to find the natural anti-log.

Example

EXP(1) returns 2.71828.

INT

INT returns the largest integer that is less than or equal to a specified value.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
INT(x)
```

Argument	Description
x	A numeric value.

Example

INT(5.6) returns 5.

INT(-5.6) returns -6.

ISUND

ISUND returns 1 if a specified value is undefined; otherwise it returns 0.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ISUND(x)
```

Argument	Description
x	A number or expression.

Example

ISUND(5.2) returns 0.

ISUND(1/0) returns 1.

LN

LN returns the natural logarithm (base e) of a number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LN(x)
```

Argument	Description
x	A positive number. The function returns an error if x is negative or zero.

Example

LN(10) returns 2.302585093.

LOG

LOG returns the base 10 logarithm of a positive number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LOG(x)
```

Argument	Description
x	A positive number. The function returns an error if x is negative or zero.

Example

LOG(10) returns 1.

MAX

MAX returns the largest number in a pair of values.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MAX(num1, num2)
```

Argument	Description
num1	The first in a pair of values.
num2	The second in a pair of values.

Example

MAX(10, 3) returns 10.

MIN

MIN returns the smallest number in a pair of values.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MIN(num1, num2)
```

Argument	Description
num1	The first in a pair of values.
num2	The second in a pair of values.

Example

MIN(10, 3) returns 3.

MOD

MOD returns the remainder of dividing a number by a divisor.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
MOD(number, divisor)
```

Argument	Description
number	The number for which you want to find the remainder.
divisor	The value by which the number argument is divided.

Example

MOD(10, 3) returns 1.

RAND

RAND generates a random number that is uniformly distributed between 0 and 1.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The random number generator is seeded when TM1 is loaded.

Syntax

```
RAND.
```

Arguments

None.

Example

RAND generates a random number that is uniformly distributed between 0 and 1

ROUND

ROUND rounds a given number to the nearest integer. Rounding can be done in a variety of valid ways.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The most basic form of rounding is to replace an arbitrary number by an integer. There are many ways of rounding a number y to an integer q.

The most common ones are:

- **Round to nearest**

q is the integer that is closest to y (see "**Round away from zero**" for tie-breaking rules).

- **Round towards zero** (or truncate)

  q is the integer part of y, without its fraction digits.
- **Round down** (or take the floor)

  q is the largest integer that does not exceed y.
- **Round up** (or take the ceiling)

  q is the smallest integer that is not less than y.
- **Round away from zero**

  If y is an integer, q is y; else q is the integer that is closest to 0 and is such that y is between 0 and q.

TurboIntegrator essentially uses the **Round down** method of *floor(x + .5)*. Microsoft Excel uses the **Round to nearest** method. This can result in different integers depending on whether you are using a TurboIntegrator process or working in Excel.

Syntax

```
ROUND(number)
```

Argument	**Description**
number	The number you want to round.

Example

ROUND(1.46) returns 1.

ROUNDP

ROUNDP rounds a given number at a specified decimal precision.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
ROUNDP(number, decimal)
```

Argument	**Description**
number	The number you want to round.
decimal	The decimal precision at which to apply the rounding. If this argument is positive, the function rounds the specified number of digits to the right of the decimal point. If this argument is negative, the function rounds the specified number of digits to the left of the decimal point. The decimal argument must be between -15 and 15, inclusive.

Example

ROUNDP(1.46, 1) returns 1.5.

ROUNDP(1.466, 2) returns 1.47.

ROUNDP(234.56, -1) returns 230.00.

ROUNDP(234.56, 0) returns 235.00.

SIGN

SIGN determines if a number is positive, negative, or zero.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The function returns 1 if the number is positive, -1 if the number is negative, and 0 if the number is zero.

Syntax

```
SIGN(number)
```

Argument	Description
number	A number.

Example

SIGN(-2.5) returns -1.

SIN

SIN returns the sine of a given angle.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SIN(x)
```

Argument	Description
x	A value, expressed in radians, for which you want the sine.

Example

SIN(1.5708) returns 1.

SQRT

SQRT returns the square root of a given value.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SQRT(x)
```

Argument	Description
x	Any positive value. SQRT returns an error if x is negative.

Example

SQRT(16) returns 4.

TAN

TAN returns the tangent of a given angle.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TAN(x)
```

Argument	Description
x	A value, expressed in radians, for which you want the tangent.

Example

TAN(0) returns 0.

TAN(.7854) returns 1.

Text Rules Functions

Rules to manage text in rules.

CAPIT

CAPIT applies initial capitalization to every word in a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CAPIT(string)
```

Argument	Description
string	A text string.

Example

CAPIT('first quarter sales') returns 'First Quarter Sales'.

CHAR

CHAR returns the character identified by a given ASCII numeric code.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CHAR(number)
```

Argument	Description
number	An ASCII code number. This number must be between 1 and 255, inclusive.

Example

CHAR(100) returns 'd'.

CODE

CODE returns the ASCII numeric code for a specified character within a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
CODE(string, location)
```

Argument	Description
string	A text string.
location	A number specifying the character within the string for which you want to find the ASCII code value.

Example

CODE('321', 2) returns 50.

CODE('End', 3) returns 100.

DELET

DELET returns the result of deleting a specified number of characters from a specified starting point within a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
DELET(string, start, number)
```

Argument	Description
string	A text string.
start	The character at which to begin deletion.
number	The number of characters to delete.

Example

DELET('payment', 3, 3) returns 'pant'.

FILL

FILL repeats a given string as necessary to return a string of a specified length.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
FILL(string, length)
```

Argument	Description
string	A text string. This string is repeated as necessary to achieve the specified length.
length	The length of the string you want the function to return.

Example

FILL('-', 5) returns '-----'.

FILL('ab', 5) returns 'ababa'.

INSRT

INSRT inserts one string into another string at a specified insertion point.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
INSRT(string1, string2, location)
```

Argument	Description
string1	A text string.
string2	A text string.
location	The character in string2 at which you want to insert string1. The function inserts string1 into string2 immediately prior to the character you specify.

Example

INSRT('ABC', 'DEF', 2) returns 'DABCEF'.

LONG

LONG returns the length of a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LONG(string)
```

Argument	Description
string	A text string.

Example

LONG('Sales') returns 5.

LOWER

LOWER converts all upper case characters in a string to lower case.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
LOWER(string)
```

Argument	Description
string	A text string.

Example

LOWER('First Quarter Sales') returns 'first quarter sales'.

NUMBR

NUMBR converts a string to a number.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The string passed to the NUMBR function must use. (period) as the decimal separator and , (comma) as the thousand separator. Any other decimal/thousand separators will cause incorrect results.

Syntax

```
NUMBR(string)
```

Argument	Description
string	The string you want to convert to a number. All characters other than '0' through '9', '+', '-', '.', and 'E' are ignored.

Example

NUMBR('-5.6') returns -5.6.

NUMBR('-5A. B6C') returns -5.6.

SCAN

SCAN returns a number indicating the starting location of the first occurrence of a specified substring within a given string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

If the substring does not occur in the given string, the function returns zero.

Syntax

```
SCAN(substring, string)
```

Argument	Description
substring	The substring you are trying to locate.
string	The string within which you are searching for the substring.

Example

SCAN('scribe', 'described') returns 3.

STR

STR converts a number to a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

The number passed to the STR function must use. (period) as the decimal separator and , (comma) as the thousand separator. Any other decimal/thousand separators will cause incorrect results.

Syntax

```
STR(number, length, decimal)
```

Argument	Description
number	The number being converted to a string.
length	The length of the string. If necessary, the function inserts leading blank spaces to attain this length.
decimal	The number of decimal places to include in the function result.

Example

STR(3.14159, 6, 2) returns ' 3.14'.

STR(-3.14159, 6, 0) returns ' -3'.

SUBST

SUBST returns a substring of a given string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
SUBST(string, beginning, length)
```

Argument	Description
string	The string from which you want to extract the substring.
beginning	The character at which the substring begins.
length	The length of the substring.

Example

SUBST('Retirement', 3, 4) returns 'tire'.

TRIM

TRIM returns the result of trimming any leading and trailing blanks from a string.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
TRIM(string)
```

Argument	Description
string	A text string.

Example

TRIM(' First Quarter ') returns 'First Quarter'.

UPPER

UPPER converts a text string to upper case.

This is a TM1 rules function, valid in both TM1 rules and TurboIntegrator processes.

Syntax

```
UPPER(string)
```

Argument	Description
string	A text string.

Example

UPPER('First Quarter Results') returns FIRST QUARTER RESULTS.

Miscellaneous Rules Functions

Rules functions not found in other categories.

FEEDERS

When you use a SKIPCHECK declaration to restore the sparse consolidation in a TM1 rule, you must also ensure that all rules-derived cells are identified by feeder statements. To do this, insert a FEEDERS declaration immediately following all rules statements:

```
FEEDERS;
```

Immediately following the FEEDERS declaration you should create feeders statements that identify the rules-derived cells in the cube.

For a complete discussion of TM1 rules, including sparse consolidation and the creation of feeders, please refer to the IBM Cognos TM1 *Rules Guide*.

FEEDSTRINGS

Rule-generated string values are not displayed when a view is zero-suppressed unless the string resides in a cell that is fed. To enable feeding of string cells, insert the FEEDSTRINGS declaration as the first line of your rule.

```
FEEDSTRINGS;
```

Once this declaration is in place, you can set up feeders for string cells in a cube view, and rely on the string to be available to other rules even if the view is zero-suppressed. Statements that define feeders for string cells should be created below the FEEDERS declaration in your rule.

As in the case of numeric feeders, a feed to a consolidated cell results in feeding of all components of the consolidation. Because you can store strings in consolidated cells, you must pay special attention if such cells are used to feed other cells. Overuse of string feeders can result in calculation explosions and poor application performance.

For a complete discussion of TM1 rules, including the creation of feeders, please refer to the IBM Cognos TM1 *Rules Guide*.

SKIPCHECK

You can restore sparse consolidation and improve performance by inserting a SKIPCHECK declaration at the beginning of the TM1 rule.

During consolidations, TM1 uses a sparse consolidation algorithm to skip over cells that contain zero or are empty. This algorithm speeds up consolidation calculations in cubes that are highly sparse. A sparse cube is a cube in which the number of populated cells as a percentage of total cells is low.

When consolidating data in cubes that have rules defined, TM1 turns off this sparse consolidation algorithm because one or more empty cells may in fact be calculated by a rule. (Skipping rules-calculated cells will cause consolidated totals to be incorrect). When the sparse consolidation algorithm is turned off, every cell is checked for a value during consolidation. This can slow down calculations in cubes that are very large and sparse.

```
SKIPCHECK;
```

If your rule uses a FEEDSTRINGS statement, the SKIPCHECK statement should be the second statement in your rule. If your rule does not use a FEEDSTRINGS statement, the SKIPCHECK statement should be the first statement in your rule.

When you use SKIPCHECK to restore sparse consolidation, you must also ensure that your rule includes a FEEDERS declaration and that all rules-derived cells are identified by feeder statements.

For a complete discussion of TM1 rules, including sparse consolidation and the creation of feeders, please refer to the IBM Cognos TM1 *Rules Guide*.

Chapter 4. Macro Functions

IBM Cognos TM1 includes a set of macro functions that you can incorporate in Excel macros. You can use macro functions to access servers, cube data and structures, and TM1 options.

Note: Before running these macros, you must load the TM1 Add-In (Tm1p.xla).

The following macro functions are described in this section:

Accessing Macro Functions

The product you are using determines the way you access TM1 macro functions.

Accessing Macro Functions from Excel Versions 5 and 7

Access macro functions from Microsoft Excel versions 5 and 7.

Procedure

1. Choose **Insert**, **Macro**, **MS Excel 4.0 Macro**.
2. Choose **Insert**, **Function**.
3. Select **TM1** from the Function Category box.
4. Double-click the function you want to insert in the Function Name list box.
5. Type the appropriate arguments in the Function Wizard dialog box.
6. Click **Finish** to copy the complete function to the current cell in the macro sheet.

Accessing Macro Functions from Excel Version 8 and Later

Access macro functions from Microsoft Excel version 8 and later.

Procedure

1. Right-click the sheet tab of the active worksheet.
2. From the shortcut menu, click **Insert**.
3. Double-click **MS Excel 4.0 Macro**.
4. Choose **Insert**, **Function**.
5. Select **TM1** from the Function category box.
6. Double-click the function you want to insert from the Function Name box.
7. Type the appropriate arguments in the Formula Palette.
8. Click **OK** to copy the complete function to the current cell in the macro sheet.

Accessing Macro Functions from VBA Modules

To access macro functions from VBA modules, use the Run method.

```
Run ("macro_function", arg1, ...)
```

Example

```
Sub Elemlist( )

Worksheets("Sheet1").Select

Cells(3,5).Select
```

```
ActiveCell.Value = Run ("E_PICK", "local:Region")
End Sub
```

This procedure calls the E_PICK macro function, which accesses a list of elements in the Region dimension. The selected element populates a cell in the Sheet1 worksheet.

D_PICK

D_PICK calls a dialog box that lists all available dimensions in the local data directory and on connected remote servers.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The dimension you select in the dialog box becomes the value of the D_PICK function.

Syntax

```
D_PICK
```

Arguments

None.

DBProportionalSpread

DBProportionalSpread distributes a specified value to the leaves of a consolidation proportional to existing cell values.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The function is analogous to the Proportional Spread data spreading method, which is described in detail in the IBM Cognos TM1 *User Guide*.

Syntax

```
DBProportionalSpread( value, server:cube, e1, e2, e3...,
e16 )
```

Argument	Description
value	The value you want to distribute.
server:cube	The name of the cube, prefixed with the appropriate server name, into which you want to distribute the value. For example, to distribute values to the Sales cube on the Accounting server, you would specify Accounting:Sales.
e1...e16	The names of the elements that identify the consolidation whose leaves will accept the distributed value.

Example

DBProportionalSpread(2000, "Accounting:Sales", "Actual", "Argentina", "S Series 1.8L Sedan", "Sales", "1 Quarter")

This example distributes the value 2000 to the children of the consolidation identified by the elements Actual, Argentina, S Series 1.8L Sedan, Sales, and 1 Quarter. It distributes values to the Sales cube on the Accounting server.

D_FSAVE

D_FSAVE lets you create or update very large dimensions whose dimension worksheets would exceed the row limit of an Excel worksheet.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

To use the D_FSAVE function, create a delimited ASCII file called dim.dit, where dim is the name of the dimension you want to create or update. This file must reside in your local server data directory.

The structure of the ASCII file must match a dimension worksheet, as follows:

- Include three fields per line.
- In the first field, specify the element type (C for consolidated; N for numeric element; S for string element; blank for consolidation component).
- In the second field, specify the element name.
- In the third field, specify the weight, if needed. The default weight is 1.0.

Separate the fields using the delimiter defined in your operating system. In Windows, this delimiter is defined by the List Separator entry in the Regional Setting Properties dialog box.

If there are errors in the structure of the ASCII file such as misplaced or undefined elements, an error message displays.

For example

Syntax

```
D_FSAVE(file)
```

Argument	Description
file	The name of a delimited ASCII file that has the file extension .dit. Do not include the file extension. This file must reside in your local TM1 data directory.

Example

=D_FSAVE("Region")

This example reads an ASCII file named Region.dit and creates or updates the Region dimension.

Note: D_FSAVE can be used to create or update dimensions on remote servers. However, the function always looks for the .dit file in the local data directory (as defined in Tm1p.ini). You must be sure that the .dit file for the dimension you want to create/update resides in your local data directory, then specify the server on which you want to create/update the dimension by prefixing the .dit file with the server name.

=D_FSAVE("TM1Serv:Region")

looks for a file named Region.dit in the local server data directory, but writes the Region dimension to the data directory for the TM1Serv server.

D_SAVE

D_SAVE saves the active worksheet as a dimension worksheet file (dim.xdi).

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The name of the workbook is used as the file name. TM1 then creates or updates the dimension specified by the workbook name.

If the active worksheet does not conform to a dimension worksheet format or is missing information, an error message displays. For example, you must define all elements used in a level-1 consolidation as numeric elements (N).

Syntax

```
D_SAVE
```

Arguments

None.

E_PICK

E_PICK calls the Subset Editor, listing all elements in the specified dimension.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The element name you select in the Subset Editor becomes the return value of the E_PICK function.

Syntax

```
E_PICK(Dimension, Alias, Subset, Element)
```

Argument	Description
Dimension	A valid dimension name. The dimension can reside in the local data directory or on a remote server to which you are connected. Use a server name prefix to indicate the server location. For the local server, specify local:dim. For a remote server, specify servername:dim.

Argument	Description
Alias	The name of an alias that exists for the subset. When this parameter is set, the alias is applied when the subset is opened in the Subset Editor and the function returns the alias for the element you select. If you choose not to set an Alias parameter you must pass an empty string to the function.
Subset	The name of the subset to be opened in the Subset Editor when E_PICK is called. The Alias parameter must be supplied to use this parameter. The Alias parameter can be defined as an empty string (""). If you choose not to set a Subset parameter you must pass an empty string to the function.
ElementNameOrIndex	The name or index number of the element to be pre-selected when the Subset Editor opens. If you choose not to set an ElementNameOrIndex parameter you must pass an empty string to the function.

Example 1

=E_PICK("TM1SERV:Region"," "," "," ")

This example opens the Region dimension in the Subset Editor.

=E_PICK ("TM1SERV:Region","Deutsch","Europe","Argentina")

This example opens the Europe subset in the Subset Editor. The Deutsche alias is applied and the Argentina element is pre-selected when the Subset Editor opens.

=E_PICK ("TM1SERV:Region"," "," ",14)

This example opens the Region dimension in the Subset Editor, with the 14th element in the dimension definition pre-selected.

I_EXPORT

I_EXPORT exports data from the specified cube to a delimited ASCII file, which is created in the current user's 'My Documents' directory. In most cases, the 'My Documents' directory is C:\Users\<user_name>\Documents.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Note: I_EXPORT applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to export a large cube, the server might be inaccessible for a significant amount of time.

Syntax

```
I_EXPORT(cube, file, zero, calcs)
```

Argument	Description
cube	A valid cube name. The cube can reside in your local data directory or on a remote server to which you are connected. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
file	The name of the delimited ASCII file to be created. The file extension .cma is used; do not specify it.
zero	Specifies whether zero values are included. Specify TRUE to include them, FALSE to exclude them.
calcs	Specifies whether calculated values are included. Specify TRUE to include them, FALSE to exclude them.

Example

=I_EXPORT("local:92act4d","Download",FALSE,TRUE)

This example exports data from the cube 92act4d to the file Download.cma. Zero values are excluded and calculated values are included.

I_NAMES

You can use I_NAMES to create a list of element names.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

This function reads through a delimited ASCII file and writes all the unique names in the specified column to the corresponding column in the active worksheet.

Syntax

```
I_NAMES(file, column)
```

Argument	Description
file	The name of an delimited ASCII file, whose file extension is .cma. Do not include the file extension.
column	A number that specifies both the field in the ASCII file from which to read names and the column in the active worksheet to which those names are written.

Example

=I_NAMES("98Sales",3)

This example inspects the file 98sales.cma. All unique names in the third column are written to column C of the active worksheet.

I_PROCESS

I_PROCESS reads in the records of an ASCII file, one at a time, into the first row of the active worksheet.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Each field populates a different cell. The worksheet is recalculated after each record is read in.

Syntax

```
I_PROCESS(file)
```

Argument	Description
file	The name of a delimited ASCII file, whose file extension is .cma. Do not include the file extension.

Example

=I_PROCESS("98Sales ")

This example reads in each record of the file 98sales.cma into the first row of the active worksheet.

M_CLEAR

M_CLEAR clears and reloads all dimensions in memory. It does not clear cubes and it does not restart the server.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Syntax

```
M_CLEAR
```

Arguments

None.

N_CONNECT

N_CONNECT establishes a connection to a remote server.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

If the connection is successful, N_CONNECT returns no value. If a connection cannot be established, server error messages are returned.

Note: The `N_CONNECT`function is not supported when a server is using Integrated Login or IBMCognos8 security for authentication. This function can only connect to a server that is configured to use standard TM1 authentication.

Syntax

```
N_CONNECT(server, client, password)
```

Argument	Description
server	The name of a remote server. This server must be registered on the Admin Server that your client references.
client	The username that connects to the specified server.
password	The password for the specified client.

Example

=N_CONNECT("Sales","USR2","Swordfish")

This example establishes a connection to the remote server named Sales, using the client name USR2 and the password Swordfish.

OPTGET

OPTGET returns the current value of an option in the Tm1p.ini file.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Syntax

```
OPTGET(option)
```

Argument	Description
option	A valid TM1 option name.

Valid Option Values	Description
AdminHost	Returns the name or address of the Admin Host your client references.
AnsiFiles	Returns T if the ANSI character set is currently used to import data from delimited ASCII files. Returns F if the ASCII character set is currently used.
DataBaseDirectory	Returns the full path to the data directory for the local server.

Valid Option Values	Description
GenDBRW	Returns F if the slice worksheet contains DBR formulas. Returns T if the slice worksheet contains DBRW formulas.
NoChangeMessage	Returns T if this option is set to return the message NO CHANGE when a DBSn formula points to a C-level cell. Returns F if this option is set to F.

Example

=OPTGET("DataBaseDirectory")

This example returns the full path to the data directory for the local server.

N_DISCONNECT

N_DISCONNECT disconnects you from all remote servers to which you are connected. The function does not disconnect you from your local server.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

N_DISCONNECT returns TRUE if it successfully disconnects you from all servers to which you are connected. It returns FALSE if cannot disconnect from any remote server.

Syntax

```
N_DISCONNECT
```

Arguments

None.

OPTSET

OPTSET sets a value for a specified TM1 option.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Syntax

```
OPTSET(option, value)
```

Argument	Description
option	A valid TM1 option name.
value	A valid value for the specified option.

Valid Option Values	Description
AdminHost	Specify the name of the Admin Host on which an Admin Server is running.
AnsiFiles	Specify a value that sets the character set used during data imports. Specify T to use the ANSI character set. Specify F to use the ASCII character set.
DataBaseDirectory	Specify a value that sets the full path to the data directory for the local server.
GenDBRW	Specify a value that determines which formula TM1 uses to link values in slice worksheets to cubes. Specify T to generate DBRW formulas when slice worksheets are created. Specify F to generate DBR formulas.
NoChangeMessage	Specify a value that determines whether TM1 displays the message NO CHANGE when a DBSn formula points to a C-level cell. Specify T to display the message. Specify F to display the value only.

Example

=OPSET("DataBaseDirectory","c:\Tm1data")

This example sets the local data directory to c:\Tm1data.

PublishSubset

PublishSubset publishes a named private subset on a server.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

If you attempt to publish a private subset for which an identically named public subset exists, you will be prompted to overwrite the existing public subset.

Syntax

```
PublishSubset(dimension, subset)
```

Argument	Description
dimension	The server-prefixed name of the dimension containing the private subset you want to publish. For example, to publish a subset of the Region dimension on the Finance server, you would pass "Finance:Region" as the dimension argument.
subset	The name of the private subset you want to publish.

PublishView

PublishView publishes a named private view on a server.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

This function cannot publish a private view that uses private subsets. All private subsets in a private view must first be published with the PublishSubset macro function.

If you attempt to publish a private view for which an identically named public view exists, you will be prompted to overwrite the existing public view.

Syntax

```
PublishView(cube, view)
```

Argument	Description
cube	The server-prefixed name of the cube containing the private view you want to publish. For example, to publish a view of the Projections cube on the Finance server, you would pass "Finance:Projections" as the cube argument.
view	The name of the private view you want to publish.

QUDEFINE

QUDEFINE sets and saves parameters for TM1 query sets.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

QUDEFINE is the equivalent of creating a query set using the View Extract dialog box.

You can run queries created with this function using the View Extract dialog box.

You can also use the query set as an argument to the QUEXPORT, QULOOP, and QUSUBSET macro functions.

Note: QUDEFINE applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to create a query that encompasses a large section of a cube, the server might be inaccessible for a significant amount of time.

Syntax

```
QUDEFINE(cube, query, range, LowLim, HiLim, SkpZeroes,
SkpCons)
```

Argument	Description
cube	The name of the cube to be queried. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
query	The name of the query set to be saved for future use.
range	A range of worksheet cells that includes one column for each dimension in the cube. When you run the query, TM1 examines only the cube cells identified by the elements specified or referenced in the range. The range must contain one column for each dimension in the cube. The order of the columns must be the same as the dimensions in the cube. In each column, you specify or reference the elements to be included. To include a subset of elements, list the element names or specify a subset name. Write the name of the subset preceded by the backslash character (\). For example, \quarter specifies the quarter subset. To include all elements in a dimension (the ALL subset), leave the column blank. You can use DBR functions to populate the cells in the range. If the functions return blank values for any column in the range, QUDEFINE uses the ALL subset for the dimension associated with that column.
LowLim	The lowest cell value to be considered for export.
HighLim	The highest cell value to be considered for export.
SkpZeroes	Specifies whether cells containing zeroes are skipped. Specify TRUE to exclude them, FALSE to include them.
SkpCons	Specifies whether cells containing consolidated values are skipped. Specify TRUE to exclude them, FALSE to include them.

Example

=QUDEFINE("local:98sales", "Topsell", Sheet1!B3:F5, 3000, 5000, TRUE, TRUE)

This example creates a query set that contains elements listed in Sheet1, in the cell range B3:F5. When you run this query, TM1 inspects only cube cells identified by these elements and exports non-consolidated values in the range 3000 to 5000.

Note: If lowlim or highlim is a string comprised of numeric characters, Excel requires the string to be enclosed in a series of four double quotation marks and single ampersands, as follows:

```
""""&"0123"&""""
```

QUDEFINEEX

QUDEFINEEX sets and saves parameters for TM1 query sets.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

It is the equivalent of creating a query set using the View Extract dialog box. This function is identical to the QUDEFINE macro, with the exception that QUDEFINEEX includes an argument that allows you to exclude rules-derived values from the query.

You can run queries created with this function using the View Extract dialog box.

You can also use the query set as an argument to the QUEXPORT, QULOOP, and QUSUBSET macro functions.

Note: QUDEFINEEX applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to create a query that encompasses a large section of a cube, the server might be inaccessible for a significant amount of time.

Syntax

```
QUDEFINEEX(cube, query, range, lowlim, hilim, skpZeroes,
skpCons, skpRuleVals)
```

Argument	Description
cube	The name of the cube to be queried.
	Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
query	The name of the query set to be saved for future use.

Argument	Description
range	A range of worksheet cells that includes one column for each dimension in the cube. When you run the query, TM1 examines only the cube cells identified by the elements specified or referenced in the range. The range must contain one column for each dimension in the cube. The order of the columns must be the same as the dimensions in the cube. In each column, you specify or reference the elements to be included. To include a subset of elements, list the element names or specify a subset name. Write the name of the subset preceded by the backslash character (\). For example, \quarter specifies the quarter subset. To include all elements in a dimension (the ALL subset), leave the column blank. You can use DBR functions to populate the cells in the range. If the functions return blank values for any column in the range, QUDEFINEEX uses the ALL subset for the dimension associated with that column.
lowlim	The lowest cell value to be considered for export.
highlim	The highest cell value to be considered for export.
skpZeroes	Specifies whether cells containing zeroes are skipped. Specify TRUE to exclude them, FALSE to include them.
skpCons	Specifies whether cells containing consolidated values are skipped. Specify TRUE to exclude them, FALSE to include them.
skpRuleVals	Specifies whether cells containing rules-derived values are skipped. Specify TRUE to exclude them, FALSE to include them.

Example

=QUDEFINEEX("local:SalesCube", "Topsell", Sheet1!B3:F5, 3000, 5000, TRUE, TRUE, FALSE)

This example creates a query set that contain elements listed in Sheet1, in the cell range B3:F5. When you run this query, TM1 inspects only cube cells identified by these elements and exports non-consolidated values in the range 3000 to 5000, including those derived through rules.

Note: If lowlim or highlim is a string comprised of numeric characters, Excel requires the string to be enclosed in a series of four double quotation marks and single ampersands, as follows:

```
""""&"0123"&""""
```

QUEXPORT

QUEXPORT exports cells values from the specified cube to a delimited ASCII file.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

To create the query set, use the QUDEFINE function.

Each output record has the following format:

- The name of the cube containing the exported values
- Names of elements that identify the cell location of a single exported value
- The exported value

For a five-dimensional cube, TM1 creates records containing seven fields:

"cube name", "elem1", "elem2", "elem3", "elem4", "elem5", value

Note: QUEXPORT applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to export values from a large query set, the server might be inaccessible for a significant amount of time.

Syntax

```
QUEXPORT(cube, query, file)
```

Argument	Description
cube	The name of the cube to be queried. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
query	The name of an existing query set.
file	The name of the delimited ASCII file (.cma) to contain the exported cube data. Do not include the file extension. The file is created in the local data directory.

Example

=QUEXPORT("sales:98sales", "Sedans", "Sedans")

This example exports data from the 98sales cube using the query set Sedans. The records are written to the file Sedans.cma.

QULOOP

QULOOP exports data that meets query set criteria from the specified cube.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

TM1 reads in each output record, one at a time, into the first row of the active worksheet. Each field populates a different cell. The worksheet is recalculated after each record is read in.

Each output record has the following format:

- The name of the cube containing the exported values
- The names of elements that identify the cell location of a single exported value
- The exported value

For a five-dimensional cube, TM1 creates records containing seven fields:

"cube name", "elem1", "elem2", "elem3", "elem4", "elem5", value

Use QULOOP in conjunction with a DBSn formula to populate cells in a cube.

Note: QULOOP applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to export values from a large query set, the server might be inaccessible for a significant amount of time.

Syntax

```
QULOOP(cube, query)
```

Argument	Description
cube	The name of the cube to be queried. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
query	The name of an existing query set.

Example

=QULOOP("sales:98sales", "Sedans")

This example exports data from the 98sales cube using the query set Sedans.

QUSUBSET

QUSUBSET is the equivalent of running a query from the View Extract dialog box when called from the Subset Editor.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Note: QUSUBSET applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to run a query that returns a large number of elements, the server might be inaccessible for a significant amount of time.

Syntax

```
QUSUBSET(cube, query, dimension, subset)
```

Argument	Description
cube	The name of the cube to be queried. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
query	The name of an existing query.
dimension	The name of a dimension for which the query exists.
subset	The name of the dimension subset to be created, which will contain the list of elements that meet the criteria of the subset. For example, a subset can return the list of regions in which car sales exceed a specified amount.

Example

=QUSUBSET("sales:98sales", "Top", "Region", "Topsales")

This example creates the Topsales subset for the Region dimension based on the criteria of the Top query.

R_SAVE

R_SAVE saves the active worksheet as a rules worksheet and compiles it into an .rux file.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The workbook must have the same name as the cube for which the rules are being compiled.

Any rules statements that prevent the rules from compiling are written to the tm1erlog.cma file, in the local data directory.

Syntax

```
RSAVE
```

Arguments

None.

SUBDEFINE

SUBDEFINE creates a dimension subset consisting of element names found in the active worksheet.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

When SUBDEFINE creates the subset, it will be created as a private subset.

If the named subset already exists as a private subset when the function is run, it will overwrite the existing private subset by that name.

If the named subset already exists as a public subset, SUBDEFINE still creates the subset as private. If you want to overwrite the existing named public subset, you will need to publish the private subset that was created by the SUBDEFINE function to overwrite the existing public subset.

Note: SUBDEFINE applies a lock to the server, preventing other users from accessing the server during function execution. If you use this function to create a subset with a large number of elements, the server might be inaccessible for a significant amount of time.

Syntax

```
SUBDEFINE(dimension, subset, range)
```

Argument	Description
dimension	The name of the dimension for which you want to create a subset. Use a server name prefix to indicate the server location. For the local server, specify local:dim. For a remote server, specify servername:dim.
subset	The name of the dimension subset.
range	The range of worksheet cells containing the names of elements in the dimension. Any cell values in the range that are not valid elements are ignored.

Example

=SUBDEFINE("local:Model", "Smith", B7:M7)

This example creates a subset called Smith for the Model dimension. The subset contains elements found in the cell range B7:M7.

SUBPICK

SUBPICK calls a dialog box that lists all the elements in the specified subset.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The elements you select are inserted in the active worksheet, starting at the current cell position.

Syntax

```
SUBPICK(dimension, subset, vertical)
```

Argument	Description
dimension	The name of the dimension containing subsets. Use a server name prefix to indicate the server location. For the local server, specify local:dim. For a remote server, specify servername:dim.
subset	The name of the subset whose elements you want to select.
vertical	Specify TRUE to insert the element names vertically, from the current cell downward. Specify FALSE to insert the element names horizontally, from the current cell rightward.

Example

=SUBPICK("local:Model", "Smith", TRUE,)

This example inserts selected elements from the Smith subset into the active worksheet. The elements are arranged vertically, starting from the current cell downward.

T_CLEAR

T_CLEAR clears all changes or additions to cube data from memory.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Note: T_CLEAR does not prompt you to save to disk any cube data in RAM. Any unsaved data is cleared without saving to disk. Therefore, if you want to save any cube data currently in RAM, call the T_SAVE function first.

Syntax

```
T_CLEAR
```

Arguments

None.

T_CREATE

T_CREATE creates a cube that has up to eight dimensions, which is the limit in older versions of TM1 .

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Note: If you use T_CREATE to create a cube with the name of an existing cube, TM1 replaces the existing cube and deletes all of its data.

Syntax

```
T_CREATE(cube,d1,d2[,d3,d4,d5,d6,d7,d8])
```

Argument	Description
cube	The name of the cube to be created. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
d1...d8	Names of up to eight existing dimensions, in the order you want them stored in the cube. You must specify at least two dimensions.

Example

=T_CREATE("local:Sales","Region","Products","Month")

This example creates a cube named Sales. This new cube has three dimensions, in the following order: Region, Products, and Month.

T_CREATE16

T_CREATE16 creates a cube that has up to sixteen dimensions.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Note: If the first argument to this function is an existing cube name, TM1 replaces the existing cube and deletes all of its data.

Syntax

```
T_CREATE16(cube,d1,d2[,d3,...,d16])
```

Argument	Description
cube	The name of the cube to be created. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
d1...d16	Names of up to sixteen existing dimensions, in the order you want them stored in the cube. You must specify at least two dimensions.

Example

=T_CREATE("Sales","Region","Products","Month")

This example creates a cube named Sales. This new cube has three dimensions, in the following order: Region, Products, and Month.

T_PICK

T_PICK calls a dialog box that lists all available cubes on the local and remote TM1 servers.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The cube name you select in the dialog box becomes the value of the T_PICK function. Your macro inserts the cube name in the first cell of the active worksheet.

Syntax

```
T_PICK
```

Arguments

None.

T_SAVE

T_SAVE saves all cube data currently in RAM to disk.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

T_SAVE can be used only to save data on a local server; the function does not work with remote servers. T_SAVE does not prompt you about saving data for individual cubes.

Syntax

```
T_SAVE
```

Arguments

None.

TM1RECALC

TM1RECALC forces a recalculation of all open worksheets. It is the equivalent of pressing F9 in Excel.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

A similar macro function, TM1RECALC1, forces a recalculation of only the active worksheet.

Syntax

```
TM1RECALC
```

Arguments

None.

TM1RECALC1

TM1RECALC1 forces a recalculation of the active worksheet. It is the equivalent of pressing SHIFT-F9 in Excel.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

A similar macro function, TM1RECALC, forces a recalculation of all open worksheets.

Syntax

```
TM1RECALC1
```

Arguments

None.

VUSLICE

VUSLICE creates a slice worksheet from the specified cube view.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

The slice is inserted starting at the top left cell (A1 or R1C1) in the active worksheet.

Syntax

```
VUSLICE(cube, view)
```

Argument	Description
cube	The name of an existing cube. Use a server name prefix to indicate the server location. For the local server, specify local:cube. For a remote server, specify servername:cube.
view	The name of a view associated with the cube.

Example

=VUSLICE("local:98sales","Quarterly")

This example copies data from the Quarterly view of the 98sales cube into the active worksheet.

W_DBSENABLE

W_DBSENABLE enables (or disables) automatic recalculation of DBS functions in a worksheet.

This is a TM1 macro function, valid only in Excel macros and VBA modules.

Normally when a DBS function is inserted in a worksheet, the function is not executed until the sheet is recalculated with either the F9 or SHIFT+F9 keys. You can use the W_DBSENABLE function to immediately execute DBS functions as they are created in a worksheet.

Note: DBS functions will not run at all in VBA modules unless W_DBSENABLE is set to TRUE.

Syntax

```
=W_DBSENABLE(LogicalFlag)
```

Argument	Description
LogicalFlag	If TRUE, DBS functions are executed immediately when inserted into or called from a worksheet. If FALSE, DBS functions are executed only when the worksheet is explicitly recalculated.

Chapter 5. Worksheet Functions

This chapter describes the functions that can be used on a worksheet.

Worksheet Function Overview

IBM Cognos TM1 Worksheet functions return a numeric or string value, and can be used anywhere in an Excel worksheet.

To access these functions in Excel, choose **Insert**, **Function** from the Excel menubar, or click fx on the Excel toolbar.

If a worksheet function references an object on a remote server, you must prefix the object with the server name and a colon. For example, to refer to the 2k2sales cube on the accounting server, use accounting:2k2sales. You must be connected to the server referenced by the function to receive accurate values in your worksheet. If you are not connected to the server, TM1 worksheet functions return *KEY_ERR.

You must adhere to the function format conventions for your spreadsheet program when using TM1 functions.

TM1 worksheet functions accept strings, values, or cell references as arguments. Strings must be enclosed in quotation marks, and cell references must refer to valid arguments for a given function. You can use standard conventions for absolute and relative cell references in worksheet functions.

If you record a worksheet macro in Excel that includes TM1 functionality, the resulting macro may include undocumented TM1 worksheet functions. We may, however, modify or discontinue these undocumented functions in future releases without notification.

Worksheet functions **cannot** be used in TM1 rules or in TurboIntegrator processes.

DBR

DBR retrieves a value from a specified TM1 cube.

This is a TM1 worksheet function, valid only in worksheets.

When all element arguments (e1, e2, etc.) to the function are leaf elements, the DBR function can also be used to write values to the specified cube, provided that the user has appropriate access privileges to the relevant cube, dimensions, elements, and/or cells. When you enter a value in a cell containing such a DBR function, the value is sent to the server.

Syntax

```
DBR(cube, e1, e2,[...en])
```

Argument	Description
cube	The name of the cube from which to retrieve the value.

Argument	Description
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements. Numeric element names must be enclosed in double quotation marks. For example ""14357"".

Example

```
DBR("92act4d", "California", "3.5 Diskettes", "Net Sales","January")
```

In this example, 92act4d is the cube name, and the function returns the value at the intersection of California, 3.5 Diskettes, Net Sales, and January.

DBRA

DBRA retrieves the value of a specified element attribute.

This is a TM1 worksheet function, valid only in worksheets.

The value returned can be either a string or numeric value, depending on the attribute type.

The DBRA function can also be used to write element attribute values to the server. When you enter a value, either string or numeric, in a cell containing a DBRA function, the corresponding element attribute is updated on the server.

Syntax

```
DBRA(server:dimension, element, attribute)
```

Argument	Description
server:dimension	A valid dimension name, prefixed with the appropriate server name and a colon, for example, "SData:Region" references the Region dimension on the SData server. If the dimension is not prefixed with a server name, the DBRA function will attempt to run against the local server.
element	An element of the dimension.
attribute	The attribute for which you want to retrieve a value. This argument must be a valid attribute of the element.

Example

```
DBRA("SData:Model", "L Series 1.8L Sedan", "ManufactureCode")
```

In this example, the function returns the value of the Manufacture Code attribute of the L Series 1.8L Sedan element in the Model dimension on the SData server.

DBRW

DBRW retrieves a value from a specified TM1 cube.

This is a TM1 worksheet function, valid only in worksheets.

When all element arguments (e1, e2, etc.) to the function are leaf elements, the DBRW function can also be used to write values to the specified cube, provided that the user has appropriate access privileges to the relevant cube, dimensions, elements, and/or cells.

This function works the same as the DBR function, with one major difference; DBRW reduces network traffic and may improve performance on wide area networks.

In worksheets with a large number of TM1 functions, DBRW forces TM1 to execute functions in "bundles" rather than individually. Normal DBR functions are executed individually during a worksheet recalculation. DBRW functions force TM1 to execute two passes over the worksheet. In the first pass, all changed values in cells containing DBRW functions are sent in a single bundle to the cube. In the second pass, cube values are sent in a single bundle back to the worksheet. Consequently, the worksheet recalculates twice when DBRW functions are executed.

DBRW bundling occurs when the function is used in a standalone cell. When DBRW functions are used in complex calculations, the function operates as a DBR function so no performance gains accrue.

Syntax

```
DBRW(cube, e1, e2[,...en])
```

Argument	Description
cube	The name of the database cube from which to retrieve the value.
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements. Numeric element names must be enclosed in quotation marks.

Example

```
DBRW("92act4d", "California", "3.5 Diskettes", "NetSales", "January")
```

In this example, the function returns the value at the intersection of California, 3.5 Diskettes, Net Sales, and January in the 92act4d cube.

DBS

DBS sends a numeric value to a TM1 cube.

This is a TM1 worksheet function, valid only in worksheets.

This function cannot send a string to a cube. To send strings, use the DBSS function.

When you build a DBS function with the **TM1** , **Edit Formula** option, the Edit Formula dialog box prompts you through a series of steps to build each function argument in the correct sequence.

If the cube does not exist or one of the arguments is invalid, the function returns KEY ERROR.

Syntax

```
DBS(value, cube, e1, e2[,...en])
```

Argument	Description
value	The value being sent.
cube	The cube to which the value is sent.
e1, ...*en*	The names of elements defining the intersection in the cube to which the value is sent. Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension of the cube, and so on. These arguments can also be the names of aliases for dimension elements. Numeric element names must be enclosed in quotation marks.

Example

```
DBS(5342,"92act4d","California","3.5 Diskettes", "NetSales", "January")
```

In this example, the function sends the value 5342 into the cube 92act4d at the intersection of California, 3.5 Diskettes, Net Sales, and January.

DBSA

DBSA sends a value to a specified element attribute.

This is a TM1 worksheet function, valid only in worksheets.

The value sent can be either a string or numeric value, depending on the attribute type.

Syntax

```
DBSA(att_value, dimension, element, att_name)
```

Argument	Description
att_value	The value you want to send.

Argument	Description
dimension	A valid dimension name. The dimension name must be prefixed with the appropriate server name and a colon, for example, "SData:Region" references the Region dimension on the SData server. If the dimension is not prefixed with a server name, the DBSA function will attempt to run against the local server.
element	An element of the dimension.
att_name	The attribute to which you want to send a value. att_name must be a valid attribute of the element specified by elem_name.

Example

```
DBSA(''LS-1.8-M7398", "SData:Model", "L Series 1.8LSedan", "Manufacture Code")
```

DBSS

DBSS sends a string to a cube of any number of dimensions.

This is a TM1 worksheet function, valid only in worksheets.

This function cannot send a numeric value to a cube. Use the DBS function to send numeric values.

When you build a DBSS function with the **TM1** , **Edit Formula** option, the Edit Formula dialog box prompts you through a series of steps to build each function argument in the correct sequence.

If the cube does not exist or one of the arguments is invalid, the function returns KEY ERROR.

Syntax

```
DBSn(string, cube, e1, e2,...en)
```

Argument	Description
string	The string being sent.
cube	The cube to which the string is sent.

Argument	Description
e1, ...e*n*	The names of elements defining the intersection in the cube to which the string is sent. Arguments e1 through en are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension of the cube, and so on. These arguments can also be the names of aliases for dimension elements.

Example

```
DBSS("Smith","Info","California","Last Name")
```

In this example, the formula sends the string Smith to the cube Info at the intersection of California and Last Name.

DBSW

DBSW sends a numeric value to a TM1 cube.

This is a TM1 worksheet function, valid only in worksheets.

This function cannot send a string to a cube. To send strings, use the DBSS function.

This function works the same as the DBS function, with one major difference; DBSW reduces network traffic and may improve performance on wide area networks.

In worksheets with a large number of cube references, DBSW forces TM1 to send values in bundles rather than individually. Normal DBS functions are updated individually during a recalculation. DBSW references force TM1 to send all changed values within a worksheet in a single bundle.

In such circumstances you can safely use a DBS/DBR function as an argument to a DBS function.

Note: If you use VBA to calculate a worksheet containing DBSW functions, you must call the TM1 macro function to calculate the worksheet. Do not use the VB Calculate method to calculate a worksheet containing DBSW functions; doing so causes each DBSW function to be executed individually, defeating the purpose of the function and resulting in decreased performance.

Syntax

```
DBSW(value, cube, e1, e2[,...en])
```

Argument	Description
value	The value being sent.

Argument	Description
cube	The cube to which the value is sent.
e1, ...e*n*	The names of elements defining the intersection in the cube to which the value is sent. Arguments e1 through en are sequence sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension of the cube, and so on. These arguments can also be the names of aliases for dimension elements. Numeric element names must be enclosed in quotation marks.

Example

```
DBSW(5342,"92act4d","California","3.5 Diskettes", "NetSales", "January")
```

DFRST

DFRST returns the first element of a specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
DFRST(server_name:dimension)
```

Argument	Description
dimension	A valid dimension name.

Example

```
DFRST("planning_sample:Location")
```

If the dimension Location contains the ordered elements California, Oregon, and Washington, the example returns California.

DIMIX

This is a TM1 worksheet function, valid only in worksheets.

DIMIX returns the index number of an element within a dimension.

Syntax

```
DIMIX(server_name:dimension, element)
```

Argument	Description
dimension	A valid dimension name.

Argument	Description
element	The name of an element within the dimension.
	If the element is not a member of the dimension specified, the function returns 0. This argument can also be the name of an alias for a dimension element.

Example

```
DIMIX("planning_sample: Location","Washington")
```

If the dimension Location contains the ordered elements California, Oregon, and Washington, the example returns the value 3, as Washington is the third element of the dimension.

DIMNM

DIMNM returns the element of a dimension that corresponds to the Index argument.

This is a TM1 worksheet function, valid only in worksheets.

If you include the optional Alias parameter to this function, the function returns the alias for the selected element

When you double-click a cell containing a DIMNM function, the Dimension dialog box opens. You can then select a new element to place in your worksheet. The DIMNM function automatically updates the index argument to reflect the new element.

Syntax

```
DIMNM(server_name:Dimension, Index, [Alias])
```

Argument	Description
Dimension	A valid dimension name.
Index	A value less than or equal to the number of elements in the dimension.
Alias	The name of an alias that exists for the dimension. This is an optional argument. If it is used, the function returns the alias for the specified element.

Example

If the Location dimension contains the ordered elements California, Oregon, and Washington, the example returns Oregon.

DIMSIZ

DIMSIZ returns the number of elements within a specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
DIMSIZ(dimension)
```

Argument	Description
dimension	A valid dimension name.

Example

```
DIMSIZ("Accounts")
```

If the Accounts dimension contains 19 elements, the example returns the value 19.

DNEXT

DNEXT returns the element name that follows the element specified as an argument to the function.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
DNEXT(server:dimension, element)
```

Argument	Description
server:dimension	A valid dimension name, prefixed with the appropriate server name and a colon, for example, "SData:Region" references the Region dimension on the SData server. If the dimension is not prefixed with a server name, the DNEXT function will attempt to run against the local server.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

```
DNEXT("Production:Location","Oregon")
```

If the Location dimension on the Production server contains the ordered elements California, Oregon, and Washington, the example returns Washington.

DNLEV

DNLEV returns the number of hierarchy levels in a dimension.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

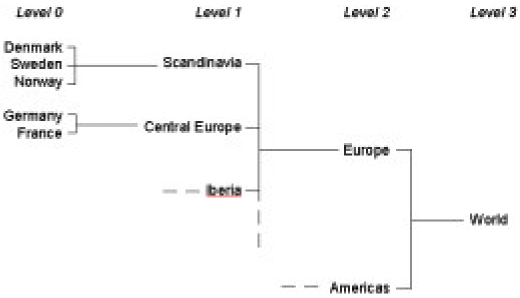
```
DNLEV(dimension)
```

Argument	Description
dimension	A valid dimension name.

Example

```
DNLEV("Region")
```

In the Region dimension, the various nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3).



In the Region dimension there are four hierarchy levels (0, 1, 2, and 3). Therefore, the example returns the value 4.

DTYPE

DTYPE returns information about the element type of the specified element. It returns "N" if the element is a numeric element, "S" if the element is a string element.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
DTYPE(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

```
DTYPE("Region","Europe")
```

The element Europe in the dimension Region is a consolidated element, so the example returns "C".

ELCOMP

ELCOMP returns the name of a child of a consolidated element in a specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

If the element argument is not a consolidated element, the function returns 0.

Syntax

```
ELCOMP(dimension, element, index)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension. This argument can also be the name of an alias for a dimension element.
index	A positive value less than or equal to the total number of children in the specified element.

Example

```
ELCOMP("Region","Central Europe",2)
```

In the dimension Region, the consolidated element Central Europe is a consolidation of the children Germany and France. Accordingly, the example returns France.

ELCOMPN

ELCOMPN returns the number of components in a specified element.

This is a TM1 worksheet function, valid only in worksheets.

If the element argument is not a consolidated element, the function returns 0.

Syntax

```
ELCOMPN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a consolidated element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

In the Region dimension, the element Scandanavia is a consolidation of three elements. The example returns 3.

ELISCOMP

ELISCOMP determines whether element1 is a child of element2 in the specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

The function returns TRUE if element1 is a child of element2, otherwise the function returns FALSE.

Syntax

```
ELISCOMP(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.
element2	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

```
ELISCOMP("Region","Germany","Central Europe")
```

In the dimension Region, the element Central Europe is a consolidation of two elements, Germany and France. The example returns TRUE.

Note that this function returns TRUE only for immediate children. In the above example, Germany is a child of Central Europe. Further, Central Europe is a child of Europe. However, because the function returns TRUE only for immediate children, the following example returns False:

```
ELISCOMP("Region","Germany","Europe")
```

ELISPAR

ELISPAR determines whether element1 is a parent of element2 in the specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

The function returns TRUE if element1 is a parent of element2, otherwise the function returns FALSE.

Syntax

```
ELISPAR(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.
element2	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

```
ELISPAR("Region","Central Europe","Germany")
```

In the dimension Region, the consolidated element Central Europe is the parent of both Germany and France. Accordingly, the example returns TRUE

Note that this function returns TRUE only for immediate parents. In the above example, Europe is a parent of Central Europe. Further, Central Europe is a parent of Germany. However, because Europe is not an immediate parent of Germany, the following example returns FALSE: `ELISPAR("Region","Europe","Germany")`

ELLEV

ELLEV returns the level of an element within a dimension.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

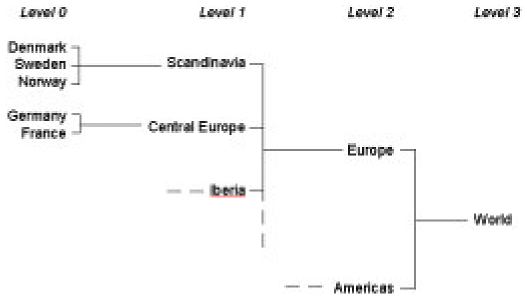
```
ELLEV(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

ELLEV("Region","Europe")

In the Region dimension, individual nations (Level 0) add up to regions (Level 1). The regions then add up to super-regions (Level 2), which in turn add up to the world (Level 3).



The example returns 2, as Europe is a Level 2 element.

ELPAR

ELPAR returns the parent of an element in a specified dimension

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
ELPAR(dimension, element, index)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.
index	A positive value less than or equal to the total number of consolidated elements (parents) that use the element argument as a child.

Example

ELPAR("Model","Wagon 4WD",2)

In the dimension Model, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. In the structure of the Model dimension, Total Wagons is defined first, Total 4WD is defined second.

The example returns Total 4WD, as this is the second instance of a parent to Wagon 4WD within the Model dimension.

ELPARN

ELPARN returns the number of parents of an element in a specified dimension.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
ELPARN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of an element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

ELPARN("Model","Wagon 4WD")

In the Model dimension, the element Wagon 4WD is a child of both Total Wagons and Total 4WD. Therefore, both Total Wagons and Total 4WD are parents of Wagon 4WD. The function returns 2.

ELSLEN

ELSLEN returns the length of a string element within a dimension.

This is a TM1 worksheet function, valid only in worksheets.

If the element specified is not a member of the dimension specified, or is not a string element, the function returns 0.

Syntax

```
ELSLEN(dimension, element)
```

Argument	Description
dimension	A valid dimension name.
element	The name of a string element within the dimension. This argument can also be the name of an alias for a dimension element.

Example

ELSLEN("Region","Washington")

The element Washington is a string element 10 characters in length. The example returns 10.

ELWEIGHT

ELWEIGHT returns the weight of a child in a consolidated element.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

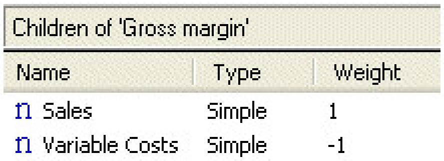
```
ELWEIGHT(dimension, element1, element2)
```

Argument	Description
dimension	A valid dimension name.
element1	The name of a consolidated element within the dimension. This argument can also be the name of an alias for a dimension element.
element2	The name of a child of the consolidated element. This argument can also be the name of an alias for a dimension element.

Example

ELWEIGHT("Account1","Gross margin","Variable costs")

As the following figure shows, the element Variable costs, which is a child of Gross margin, has a weight of -1.



Children of 'Gross margin'

Name	Type	Weight
Sales	Simple	1
Variable Costs	Simple	-1

The example returns -1.

SUBNM

SUBNM returns the element of a dimension subset corresponding to the IndexOrName argument.

This is a TM1 worksheet function, valid only in worksheets.

When you double-click a cell containing a SUBNM function, the Subset Editor opens. You can then select a new element to place in your worksheet. The selected element becomes the return value of the SUBNM function, and the function automatically updates the IndexOrName argument to reflect the new element.

If you include the optional Alias parameter to this function, the function returns the alias for the selected element

Syntax

```
SUBNM(Dimension, Subset, IndexOrName, [Alias])
```

Argument	Description
Dimension	A valid dimension name.

Argument	Description
Subset	The name of a subset of the dimension.
IndexOrName	An index into the subset or the name of an element in the subset. If an index, a positive integer less than or equal to the total number of elements in the specified subset. If a name, a string representing the name of an element of the subset.
Alias	The name of an alias that exists for the subset. This is an optional argument. If it is used, the specified alias is applied when the Subset Editor opens and the function returns the alias for the selected element.

Example

SUBNM("Region","Top Producers",2)

The Top Producers subset of the Region dimension contains the ordered elements United States, Germany, Great Britain, and Mexico. Because the Index argument points to the second element in the subset, the example returns Germany.

SUBNM("Region","Top Producers","Germany","Deutsch")

This example returns the Deutsch alias for the Germany element (Deutschland) from the Top Producers subset of the Region dimension.

SUBSIZ

SUBSIZ returns the number of elements in a dimension subset.

This is a TM1 worksheet function, valid only in worksheets.

Syntax

```
SUBSIZ(dimension, subset)
```

Argument	Description
dimension	A valid dimension name.
subset	The name of a subset of the dimension.

Example

SUBSIZ("Region","Top Producers")

The Top Producers subset of the Region dimension contains four elements: United States, Germany, Great Britain, and Mexico.

The example returns 4.

TABDIM

TABDIM returns the dimension name that corresponds to a given index argument.

This is a TM1 worksheet function, valid only in worksheets.

The function always returns a dimension based on the original order of dimensions in the specified cube, even if the order of dimensions in the cube has been changed through the TM1 Cube Optimizer.

Syntax

```
TABDIM(cube, index)
```

Argument	Description
cube	A valid cube name.
index	A positive value less than or equal to the total number of dimensions in the cube.

Example

TABDIM("98sales",3)

The cube 98sales contains five dimensions: account1, actvsbud, model, month, and region. The example returns model, the third dimension of 98sales.

TM1RptElIsConsolidated

TM1RptElIsConsolidated returns a Boolean value to indicate whether an element in an Active Form is consolidated.

This is a TM1 worksheet function, used to create Active Forms.

Syntax

```
TM1RptElIsConsolidated(RptRowFormula, Element)
```

Argument	Description
RptRowFormula	An absolute reference to a cell containing a TM1RptRow formula.
Element	A relative reference to a cell containing an element from TM1RptRow formula.

TM1RptElIsExpanded

TM1RptElIsExpanded returns a boolean value to indicate whether an element is expanded in a row subset within an Active Form.

This is a TM1 worksheet function, used to create Active Forms.

Syntax

```
TM1RptElIsExpanded(RptRowFormula, Element)
```

Argument	Description
RptRowFormula	An absolute reference to a cell containing a TM1RptRow formula.
Element	A relative reference to a cell containing an element from TM1RptRow formula.

TM1RptElLev

TM1RptElLev returns an integer value for an element level relative to root in the subset.

This is a TM1 worksheet function, used to create Active Forms.

This function is distinct from the ElLev worksheet function.

Syntax

```
TM1RptElLev(RptRowFormula, Element)
```

Argument	Description
RptRowFormula	An absolute reference to a TM1RptRow formula cell.
Element	A relative reference to a cell containing an element from TM1RptRow formula.

TM1RptFilter

TM1RptFilter defines the filter applied to an Active Form column dimension.

This is a TM1 worksheet function, used to create Active Forms.

Syntax

```
TM1RptFilter(ReportView,Tuple,FilterFunction,FilterValue,SortOrder)
```

Argument	Description
ReportView	A cell reference to a cell that contains a TM1RptView formula. The filter applies to the view specified by TM1RptView formula.
Tuple	A tuple string specifying the element in the column dimension to which the filter applies. For example, [month].[Feb].

Argument	Description
FilterFunction	One of the following filter function names: TOPCOUNT BOTTOMCOUNT TOPPERCENT BOTTOMPERCENT TOPSUM BOTTOMSUM
FilterValue	A filter value.
SortOrder	One of the following two sort orders: asc desc

Example

=TM1RptFilter(B4,"[month].[Jan]","TOPCOUNT",5,"asc")

TM1RptRow

TM1RptRow sets the Active Form master row definition.

This is a TM1 worksheet function, used to create Active Forms.

The master row definition governs the behavior of all rows in the Active Form.

Syntax

```
TM1RptRow(ReportView, Dimension, Subset, SubsetElements,
Alias, ExpandAbove,MDXStatement, Indentations, ConsolidationDrilling)
```

Argument	Description
ReportView	A reference to a cell that contains a TM1RptView formula.
Dimension	A dimension, specified using the format TM1 _server_name:dimension_name.
Subset	A named subset. If this argument is empty, all elements of the dimension will be used.

Argument	Description
SubsetElements	A cell range reference that specifies a list of elements to constitute a subset. When this argument is supplied, the named subset specified by the Subset argument is ignored. If this argument is empty, the elements from the subset specified by the Subset argument are used.
Alias	A string that defines the alias used for the subset. When this argument is supplied, it overrides the default alias property defined by the subset specified by the Subset argument. If this argument is empty, the alias from the subset specified by the Subset argument are used.
ExpandAbove	A Boolean flag to turn on or off the subset Expand Above property. When this argument is supplied, it overrides the default Expand Above property defined by the subset specified by the Subset argument. If the argument value is 1, consolidated elements expand upward when drilling. If the argument value is 0, consolidated elements expand downward when drilling. If this argument is empty, the Expand Above property from the subset specified by the Subset argument is used.
MDXStatement	An MDX statement that applies to the subset specified by the Subset argument. When this argument is supplied, it overrides the default MDX filter defined by the subset specified by the Subset argument. If this argument is empty or omitted, the elements from the subset specified by the Subset argument are used.

Argument	Description
Indentations	An integer value to indicate how many indentations are applied to each level when drilling down on a consolidated element. If the argument value is 0, no auto-indentation is performed. This is an optional argument. When the value is missing, one indentation is applied to each level as you drill down on a consolidated element.
ConsolidationDrilling	A Boolean flag to turn on or off drilling on consolidated elements. When this argument value is 1, users can drill down on consolidated elements in the Active Form. When this argument value is 0, users can not drill down on consolidated elements in the Active Form. This is an optional argument. When the argument is missing, the default behavior is to allow drilling on consolidated elements.

Example

=TM1RptRow($B$9,"sdata:region","",'{AR}01'!$B$17:$B$18,"",1,"",5, 0)

TM1RptTitle

TM1RptTitle defines an Active Form title dimension.

This is a TM1 worksheet function, used to create Active Forms.

Syntax

```
TM1RptTitle(Dimension,Element)
```

Argument	Description
Dimension	A dimension, specified using the format TM1 _name:dimension_name.
Element	A cell reference to a cell containing a SUBNM function which returns an element name.

Example

TM1RptTitle("SData:model",$C$7)

TM1RptView

TM1RptView defines the view displayed in an Active Form.

This is a TM1 worksheet function, used to create Active Forms.

Syntax

```
TM1RptView(ViewID,ZeroSuppression,TM1RptTitle,...)
```

Argument	Description
ViewID	A name for the view using the format TM1 _name:cube_name:unique_id.
ZeroSuppressio	A Boolean flag to turn on or off the zero suppression property for the view. 1 = on, 0 = off
TM1RptTitle	For each title dimension in the Active Form, include a TM1RptTitle function as an argument to TM1RptView.
FormatRange	The formatting range for the Active Form. When you create an Active Form, a named range called TM1RPTFMTRNG is created to include all formatting range cells. You can use this named range as an argument.
IDColumn	The column containing format IDs in the Active Form. When you create an Active Form, a named range called TM1RPTFMTIDCOL is created to include all formatting range cells. You can use this named range as an argument.

Example

=TM1RPTVIEW("SData:SalesCube:6", 0, TM1RPTTITLE("SData:actvsbud",C6), TM1RPTTITLE("SData:model",C7), TM1RPTTITLE("SData:account1",C8), TM1RPTFMTRNG,TM1RPTFMTIDCOL)

TM1User

TM1User returns the user name of the current TM1 user.

This is a TM1 worksheet function, valid only in worksheets.

If the current TM1 user is not connected to a server, or if the specified server is not running, TM1User returns an empty string.

If TM1User is executed against a server that is configured to use CAM authentication, the function returns the internal user name/CAMID, not the display name.

Syntax

```
TM1User("ServerName")
```

Argument	Description
ServerName	The name of the server to which the TM1 user is connected.

Example

TM1User("SData")

If a user named BrianT is logged in to the SData server, and that user executes the TM1User function, the above example returns BrianT.

VIEW

VIEW creates an optimized view of the cube specified by the cube argument.

This is a TM1 worksheet function, valid only in worksheets.

A single VIEW function is created when you slice a view from a cube browse.

All DBR and DBRW formulas that refer to the VIEW function can then access this optimized view. In this way, results are returned much faster.

Multiple VIEW functions can reside in the same spreadsheet if you have blocks of DBR formulas that refer to different TM1 views and/or cubes.

Syntax

```
VIEW(cube, e1,e2[,...en])
```

Argument	Description
cube	The name of the cube from which to retrieve data.
e1,...*en*	Either specific elements in the slice to be used as titles, or the string "!". The string "!" indicates that the corresponding dimension is a row or column in the view. These arguments can also be the names of aliases for dimension elements.

Example

VIEW("93sales",$B$2,$B$3,$B$4,"!","!")

Chapter 6. TM1 TurboIntegrator Functions

TM1 TurboIntegrator lets you manipulate TM1 data and metadata when you define a process.

This is accomplished through the use of functions in the Prolog, Metadata, Data, and Epilog sub-tabs within the Advanced tab of the TurboIntegrator window. These sub-tabs include generated statements based on settings and options you select when defining a TurboIntegrator process. Any functions you create must appear after the generated statements. For details on creating processes with TurboIntegrator, see the *IBM CognosTM1 TurboIntegrator Guide*.

The TurboIntegrator functions in this section are sorted by category.

There is no interface to assist in the creation of TurboIntegrator functions. Enter functions by hand directly in the appropriate sub-tab within the Advanced tab. String arguments to TurboIntegrator functions must be enclosed in single quotation marks. A semi-colon (;) must be included to indicate the end of each function in the TurboIntegrator window.

In addition to these TurboIntegrator functions, you can also incorporate all standard TM1 Rules functions in a process definition, with the exception of the STET function.

Each argument to TurboIntegrator functions is limited to 256 bytes. A TurboIntegrator function can accept multiple arguments, and each argument is limited to 256 bytes.

ASCII and Text TurboIntegrator Functions

These functions pertain to ASCII and Text.

ASCIIDelete

ASCIIDelete deletes an ASCII file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ASCIIDelete(FileName);
```

FileName

The name of the ASCII file you want to delete. If a full path is not specified, TM1 searches for the file in the server data directory.

Example

```
ASCIIDelete('C:\exported_data\2002Q1Results.cma');
```

This example deletes the ASCII file named 2002Q1Results.cma from the C:\exported_data directory.

ASCIIOutput

ASCIIOutput writes a comma-delimited record to an ASCII file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The ASCII file is opened when the first record is written, and is closed when the TurboIntegrator procedure (Prolog, Metadata, Data, or Epilog) containing the ASCIIIOutput function finishes processing.

Each output record generated by ASCIIOutput is limited to 8000 bytes. If an output record exceeds 8000 bytes, the record is truncated and a warning is logged in the TM1ProcessError.log file.

When ASCIIOutput encounters a String argument that pushes the output record beyond the 8000 byte limit, it ignores that argument and any further arguments. For example, if there are 10 String arguments and output for the first seven arguments total 7950 bytes while the output for the eighth argument is 51 bytes, only the output for the first seven arguments will be written to the record. If there are ten String arguments and the first argument is over 8000 bytes, no output will be written to the record.

If you use the ASCIIOutput function to write to the same file in multiple procedures (tabs) of a TurboIntegrator process, the file will be overwritten each time it is opened for a new procedure.

The ASCIIOutput function generates a minor error if an error occurs while writing the ASCII file. In addition, the function returns a value upon execution: 1 if the function successfully writes the ASCII file and 0 on failure.

Note that the error will be generated and the value returned only when ASCIIOutput is writing to a disk other than the one that the server is running on. For example, if the server is running on the C: drive and ASCIIOutput is writing to the F: drive, and the F: drive runs out of space, the error will be trapped and the server remains alive. If the server is running on the C: drive while ASCIIOutput is also writing to the C: drive, and that drive runs out of space, the server will terminate (as expected).

Note: The ASCIIOutput function places the 0x1A hexadecimal character at the end of all generated files. However, TM1 Web cannot open a Websheet that contains the 0x1A hexadecimal character.

If you use ASCIIOutput to export TM1 data to an ASCII file and then attempt to open the file in a TM1 Websheet, you will encounter the following error.

Error occurred while converting the MS Excel workbook into XML format, hexadecimal value 0x1A is an invalid character.

If you remove the 0x1A hexadecimal character from the Websheet, the file will open in TM1 Web.

Syntax

```
ASCIIOutput(FileName, String1, String2, ...Stringn);
```

Argument	Description
FileName	A full path to the ASCII file to which you want to write the record. Path must include a file extension.
String1...String*n*	A string that corresponds to each field you want to create in the ASCII file. This argument can be a string or a TurboIntegrator variable for a string.

Example

```
ASCIIOutput('NewCube.cma', V1, V2, V3, V4, V5 );
```

This example writes a record to the NewCube.cma ASCII file. Each field in the record corresponds to a variable assigned by TurboIntegrator to a column in your data source.

SetInputCharacterSet

SetInputCharacterSet function lets you specify the character set used in a TurboIntegrator data source.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

When a TurboIntegrator process reads an external file as input, it needs to know the character set in which that external file was written. If the file contains a valid byte-order-mark, TM1 functions will correctly convert the file to UTF-8 if required.

For formats lacking a valid byte-order-mark, the characters must be converted from some other encoding to UTF-8. If the proper converters are present on the machine hosting the server, the input file will be converted to the Unicode character set required by TM1 .

Syntax

```
SetInputCharacterSet (CharacterSet);
```

Argument	Description
CharacterSet	The character encoding in the input file to be used by the TurboIntegrator process. If the CharacterSet argument is not a known character type, the type defaults to the system locale.

Character Encoding	System Locale
TM1CS_ISO_8859_1	ISO-8859-1 Latin-1, Western Europe

Character Encoding	System Locale
TM1CS_ISO_8859_2	ISO-8859-2 Latin-2, Central Europe
TM1CS_ISO_8859_3	ISO-8859-3 Latin-3, South Europe
TM1CS_ISO_8859_4	ISO-8859-4 Latin-4, North Europe
TM1CS_ISO_8859_5	ISO-8859-5 Latin/Cyrillic
TM1CS_ISO_8859_6	ISO-8859-6 Latin/Arabic
TM1CS_ISO_8859_7	ISO-8859-7 Latin/Greek
TM1CS_ISO_8859_8	ISO-8859-8 Latin/Hebrew
TM1CS_ISO_8859_9	ISO-8859-9 Latin-5, Turkish
TM1CS_ISO_8859_10	ISO-8859-10 Latin-6, Nordic,
TM1CS_ISO_8859_11	ISO-8859-11 Latin/Thai
TM1CS_ISO_8859_13	ISO-8859-13 Latin-7, Baltic Rim
TM1CS_ISO_8859_14	ISO-8859-14 Latin-8, Celtic
TM1CS_ISO_8859_15	ISO-8859-15 Latin-9, replaces ISO-8859-1
TM1CS_ISO_8859_16	ISO-8859-16 Latin-10, South-Eastern Europe
TM1CS_WCP1250	Microsoft Windows Central Europe
TM1CS_WCP1251	Windows Cyrillic
TM1CS_WCP1252	Windows Latin-1 multilingual
TM1CS_WCP1253	Windows Greek
TM1CS_WCP1254	Windows Turkish
TM1CS_WCP1255	Windows Hebrew
TM1CS_WCP1256	Windows Arabic
TM1CS_WCP1257	Windows Baltic
TM1CS_WCP1258	Windows Vietnam
TM1CS_WCP874	Windows Thai
TM1CS_WCP932	Windows Japanese

Character Encoding	System Locale
TM1CS_WCP936	Windows Simplified Chinese
TM1CS_WCP949	Windows Korean
TM1CS_WCP950	Windows Traditional Chinese
TM1CS_KOI8R	Russian and Cyrillic (KOI8-R)
TM1CS_GB18030	PRC version UNICODE
TM1CS_BIG5	Traditional Chinese
TM1CS_SHIFTJIS	JIS 0201 + JIS 0208, slightly different from CP932
TM1CS_SJIS0213	JIS 0213-2004, non-BMP required.
TM1CS_EUC_JP	EUC Japanese
TM1CS_EUC_CN	EUC Simplified Chinese
TM1CS_EUC_KR	EUC Korean
TM1CS_UTF8	UTF-8
TM1CS_UTF16	UTF-16 Little Endian
TM1CS_UTF16ESC	UNICODE notation
TM1CS_UTF32	UTF-32 Little Endian
TM1CS_OS_DEFAULT	operating system default
TM1CS_LOCALPATH	local encoding but UNICODE notation on non-native.

Example

```
SetInputCharacterSet ('TM1CS_ISO_8859_11');
```

This example specifies that the input character set for the TurboIntegrator data source is ISO-8859-11 Latin/Thai.

SetOutputCharacterSet

SetOutputCharacterSet lets you specify the character set to be used when writing to a text file using TextOutput in a TurboIntegrator proces.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The function with the TextOutput function.

Used withTextOutputs.

Syntax

```
SetOutputCharacterSet( FileName, CharacterSet );
```

Argument	Description
FileName	A full path to the text file for which you want to specify a character set. The path must include a file extension. This argument should be indentical to the FileName argument for the TextOutput function.
CharacterSet	The character encoding to use when writing to the output file.

For more information on the valid values for CharacterSet, see "SetInputCharacterSet" on page 203.

SetOutputEscapeDoubleQuote

SetOutputEscapeDoubleQuote allows you to escape double quotes that appear in element names or data values when exporting a cube view to a .csv file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

When SetOutputEscapeDoubleQuote is included in your TurboIntegrator script and set to 1, the exported file retains the double quote positions as they appear in your source cube view by escaping each double quote within another pair of double quotes. For example, if an element in your source view is named "Region", the element is exported as """Region""" in the .csv output file.

When SetOutputEscapeDoubleQuote is *not* included in your TurboIntegrator script or is set to 0, the exported file does not escape any double quotes that appear in your source cube.

SetOutputEscapeDoubleQuote is used in conjunction with the ASCIIOutput function, which is the function that actually writes the output file. SetOutputEscapeDoubleQuote should precede ASCIIOutput in your TurboIntegrator script, and both functions should use the same FileName parameter value.

Syntax

```
SetOutputEscapeDoubleQuote(FileName, Num);
```

Argument	Description
FileName	A full path to the file to which you want to write the cube view. Path must include a file extension.

Argument	Description
Num	A flag that determines if double quotes are escaped in the output file. 1 indicates that double quotes will be escaped in the output file. 0 indicates that double quotes will not be escaped in the output file.

Example

```
SetOutputEscapeDoubleQuote('C:\temp\cube1.csv', 1);
```

This example escapes any double quotes encountered in the source cube view when writing output to the C:\temp\cube1.csv file.

TextOutput

TextOutput writes a comma-delimited record to a text file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

By default TextOutput writes characters in the locale character set of the server machine. To create a file in a different character set, call the function SetOutputCharacterSetbefore calling TextOutput.

The text file is opened when the first record is written, and is closed when the TurboIntegrator procedure (Prolog, Metadata, Data, or Epilog) containing the TextOutput function finishes processing.

If you use the TextOutput function to write to the same file in multiple procedures (tabs) of a TurboIntegrator process, the file will be overwritten each time it is opened for a new procedure.

Each output record generated by TextOutput is limited to 8000 bytes. If an output record exceeds 8000 bytes, the record is truncated and a warning is logged in the TM1ProcessError.log file.

When TextOutput encounters a String argument that pushes the output record beyond the 8000 byte limit, it ignores that argument and any further arguments. For example, if there are 10 String arguments and output for the first seven arguments total 7950 bytes while the output for the eighth argument is 51 bytes, only the output for the first seven arguments will be written to the record. If there are ten String arguments and the first argument is over 8000 bytes, no output will be written to the record.

The TextOutput function generates a minor error if an error occurs while writing the text file. In addition, the function returns a value upon execution: 1 if the function successfully writes the text file and 0 on failure.

The error will be generated and the value returned only when TextOutput is writing to a disk other than the one that the server is running on. For example, if the server is running on the C: drive and TextOutput is writing to the F: drive, and the F: drive runs out of space, the error will be trapped and the server remains alive. If the server is running on the C: drive while TextOutput is also writing to the C: drive, and that drive runs out of space, the server will terminate (as expected).

Syntax

```
TextOutput(FileName, String1, String2, ...Stringn);
```

Argument	Description
FileName	A full path to the text file to which you want to write the record. Path must include a file extension.
String1...String*n*	A string that corresponds to each field you want to create in the text file. This argument can be a string or a TurboIntegrator variable for a string.

Example

```
TextOutput('NewCube.cma', V1, V2, V3, V4, V5 );
```

This example writes a record to the NewCube.cma file. Each field in the record corresponds to a variable assigned by TurboIntegrator to a column in your data source.

Attribute Manipulation TurboIntegrator Functions

These functions facilitate the manipulation of attributes.

AttrDelete

AttrDelete deletes an element attribute from the TM1 database.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
AttrDelete(DimName, AttrName);
```

Argument	Description
DimName	The dimension for which you want to delete an element attribute.
AttrName	The name of the attribute you want to delete.

Example

```
AttrDelete('Model', 'InteriorColor');
```

This example deletes the InteriorColor element attribute for the Model dimension.

AttrInsert

AttrInsert creates a new element attribute for a dimension. The function can create a string, numeric, or alias attribute.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
AttrInsert(DimName, PrevAttr, AttrName, Type);
```

Argument	Description
DimName	The dimension for which you want to create an element attribute.
PrevAttr	The attribute that precedes the attribute you are creating.
AttrName	The name you want to assign to the new attribute.
Type	The type of attribute. There are three possible values for the Type argument: • N - Creates a numeric attribute. • S - Creates a string attribute. • A - Creates an alias attribute.

Example

```
AttrInsert('Model', 'Transmission', 'InteriorColor','S');
```

This example creates the InteriorColor string attribute for the Model dimension. This attribute is inserted after the Transmission attribute.

AttrPutN

AttrPutN assigns a value to a numeric element attribute.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
AttrPutN(Value, DimName, ElName, AttrName);
```

Argument	Description
Value	The value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.
ElName	The element for which you want to assign an attribute value.

Argument	Description
AttrName	The attribute whose value you want to assign.

Example

```
AttrPutN(2257993, 'Model', ' S Series 1.8L Sedan ','ProdCode');
```

This example assigns the value 2257993 to the ProdCode attribute of the S Series 1.8L Sedan in the Model dimension.

AttrPutS

AttrPutS assigns a value to a string element attribute.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
AttrPutS(Value, DimName, ElName, AttrName);
```

Argument	Description
Value	The value you want to assign to an element attribute.
DimName	The parent dimension of the element for which you want to assign an attribute value.
ElName	The element for which you want to assign an attribute value.
AttrName	The attribute whose value you want to assign.

Example

```
AttrPutS('Beige', 'Model', 'S Series 1.8L Sedan', 'InteriorColor');
```

This example assigns the string Beige to the InteriorColor attribute of the S Series 1.8L Sedan in the Model dimension.

Chore Management TurboIntegrator Functions

These functions pertain to managing chores.

ChoreQuit

ChoreQuit causes the immediate termination of a chore. It can be called from any process within a chore.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

When a process encounters the ChoreQuit function, the current chore is terminated with an error status, and a message is written to the server log file indicating that ChoreQuit was called to terminate the chore.

Syntax

```
ChoreQuit;
```

Arguments

None.

SetChoreVerboseMessages

Use SetChoreVerboseMessages to turn on (or off) more verbose reporting of messages to the Tm1s.log file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function is best used as an aid to debugging chores in which several processes call one another through use of the ExecuteProcess function.

Passing a zero value turns off the output of these messages, passing a non-zero value enables the output of more verbose messages. By default this flag is off.

Syntax

Use this function to turn on (or off) more verbose reporting of messages to the Tm1s.log file. This function is best used as an aid to debugging chores in which several processes call one another through use of the ExecuteProcess function.

Passing a zero value turns off the output of these messages, passing a non-zero value enables the output of more verbose messages. By default this flag is off.

```
SetChoreVerboseMessages(Flag);
```

Argument	Description
Flag	Set to a non-zero value to enable more verbose messaging. Set to zero (default) to turn off verbose messaging.

Cube Manipulation TurboIntegrator Functions

These functions pertain to manipulating cubes.

AddCubeDependency

AddCubeDependency lets you pre-define cube interdependencies to avoid lock contention problems during normal system use.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

In normal operations, cube dependencies are established when data which crosses cube boundaries (such as data that is derived by a rule that references an external cube) is retrieved. To create the dependency information, the server must lock the cubes while the dependency is established, potentially maintaining the lock during a long view calculation. Since this is a 'write' lock, other users are prevented from accessing the cubes. The AddCubeDependency function allows the dependency to be established when the server starts up, preventing later lock contention as no new dependency need be established.

Syntax

```
AddCubeDependency(BaseCube, DependentCube);
```

Argument	Description
BaseCube	The name of a cube that depends on another cube for data. Most commonly, this would be a cube that uses rules to pull data from an external cube.
DependentCube	The name of the cube upon which the BaseCube is dependent.

Example

```
AddCubeDependency('A1', 'A2');
```

Consider two cubes, A1 and A2. Cube A1 has this rule for Jan-03 which pulls data from cube A2: `['Jan-03']=DB('A2', elem_1, elem_2 , ...);`

In this case A1 is dependent on A2. To establish the dependency in TurboIntegrator you would include the example shown above.

CellGetN

This function retrieves a value from a numeric cube cell.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellGetN(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube from which you want to retrieve a value.

Argument	Description
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables. **Note:** : When this function is used in a conditional statement (IF), the statement is the portion containing the condition, not the entire conditional block. After a minor error, execution continues with the next statement. TI processing has no knowledge that it was in a conditional once the minor error is processed, so the next statement is the next line, not the line after the endif. To avoid this situation, use variables for any operation that could encounter a minor error and then use the variables in the conditional statement. For example:

`V1 = CELLGETN('PNLCube', 'fred','argentina','Sales','Jan');IF(V1 = 454);`
`ASCIIOUTPUT('bug.txt', 'if logic not working properly');ENDIF;`	

Example

```
CellGetN ('y2ksales', 'Actual', 'Argentina', 'S Series1.8L Sedan', 'Sales', 'Jan');
```

This example retrieves the numeric value at the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the y2ksales cube.

CellGetS

This function retrieves a value from a string cube cell.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellGetS(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube from which you want to retrieve a value.

Argument	Description
e1,...e*n*	Dimension element names that define the intersection of the cube containing the value to be retrieved. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables. See the note at “CellGetN” on page 212concerning IF logic with this function.

Example

```
CellGetS('Personnel', 'Rep', 'Europe', 'Product');
```

This example retrieves the string value at the intersection of the Rep, Europe, and Product elements in the Personnel cube.

CellIncrementN

This function increments an existing numeric cell value by a specified value.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellIncrementN(x, Cube, e1, e2 [,...en]);
```

Argument	Description
x	A numeric value that you want to add to an existing cell value.
Cube	The name of the cube to which you want to send the value.
e1,...e*n*	Dimension element names that define the intersection of the cube to receive the value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellIncrementN(1000, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example increments the value at the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the y2ksales cube by 1000.

CellIsUpdateable

This function lets you determine if a cube cell can be written to. The function returns 1 if the cell can be written to, otherwise it returns 0.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellIsUpdateable(Cube, e1, e2 [,...en]);
```

Argument	Description
Cube	The name of the cube to which you want to write a value.
e1,...e*n*	Dimension element names that define the cell to which you want to write a value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellIsUpdateable ('y2ksales', 'Actual', 'Argentina','S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example determines if the cell defined by the elements Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan in the y2ksales cube can be written to. If the cell can receive a value, the function returns 1, otherwise it returns 0.

CellPutN

This function sends a numeric value to a cube cell.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellPutN(x, Cube, e1, e2 [,...en]);
```

Argument	Description
x	A numeric value.
Cube	The name of the cube to which you want to send the value.

Argument	Description
e1,...e*n*	Dimension element names that define the intersection of the cube to receive the value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellPutN(12345, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8L Sedan', 'Sales', 'Jan');
```

This example sends the value 12345 to the intersection of the Actual, Argentina, S Series 1.8L Sedan, Sales, and Jan elements in the y2ksales cube.

CellPutProportionalSpread

This function distributes a specified value to the leaves of a consolidation proportional to existing cell values.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

CellPutProportionalSpread replaces existing cell values; it cannot be used to add to or subtract from existing cell values.

The function is analogous to the Proportional Spread data spreading method, which is described in detail in the IBM Cognos TM1 *User Guide*. If you must add to or subtract from existing cell values, use the Proportional Spread method, which can be executed through the user interface or through data spreading syntax.

Note: When using CellPutProportionalSpread to distribute a value to the leaves of a consolidation, only those leaves already containing non-zero values are changed. This is because zero values cannot be incremented or decremented proportionally; any proportion of zero is still zero.

Syntax

```
CellPutProportionalSpread( value, cube, e1, e2, e3...,en );
```

Argument	Description
value	The value you want to distribute.
cube	The name of the cube into which you want to distribute the value.

Argument	Description
e1...e*n*	The names of the elements that identify the consolidation whose leaves will accept the distributed value. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellPutProportionalSpread(7000,'SalesCube', 'Actual','North America',
'S Series 1.8L Sedan', 'Sales', 'Jan')
```

This example distributes the value 7000 to the children of the consolidation in the SalesCube identified by the elements Actual, North America, S Series 1.8L Sedan, Sales, and Jan.

CellPutS

This function sends a string value to a cube cell.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CellPutS(String, Cube, e1, e2 [,...en]);
```

Argument	Description
String	A string.
Cube	The name of the cube to which you want to send the string.
e1,...e*n*	Dimension element names that define the intersection of the cube to receive the string. Arguments e1 through e*n* are sequence-sensitive. e1 must be an element from the first dimension of the cube, e2 must be an element from the second dimension, and so on. These arguments can also be the names of aliases for dimension elements or TurboIntegrator variables.

Example

```
CellPutS('jones', 'Personnel', 'Rep', 'Europe', 'Product');
```

This example sends the string 'jones' to the intersection of the Rep, Europe, and Product elements in the personnel cube.

CubeClearData

This clears all of the data in a cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function is much faster than doing an operation such as creating a view to cover the entire cube, and then doing a ViewZeroOut() to zero out the entire cube.

When you use CubeClearData to clear data from a cube, any cells in the cube that are fed with feeders are also cleared. You must resave the rule that establishes the feeders or use the CubeProcessFeeders function to restore the fed cells.

Note: This call just deletes the cube data, it does not delete and re-create the cube itself. This has implications when sandboxes are used. If a cube is deleted and then re-created any sandboxes a user may have will be discarded, since the cube against which those sandboxes were created was deleted (even though a cube may have been re-created with the same name). If however the CubeClearData() call is used, the sandbox data will still be considered valid, since the cube against which the sandbox was created continues to exist.

Syntax

```
CubeClearData( name-of-cube-as-string );
```

Argument

The name of the cube to clear, as a string.

Example

```
CubeClearData( 'expense' );
```

CubeCreate

This function creates a cube from specified dimensions. The order of dimensions specified in the function will be the order of dimensions in the cube definition.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

After execution, CubeCreate automatically saves the resulting .cub file to disk.

Syntax

```
CubeCreate(Cube, d1, d2 [,...dn]);
```

Argument	Description
Cube	The name you want to assign to the cube.
d1,...d*n*	The names of dimensions that comprise the cube. You must specify at least two, but no more than 256, dimensions.

Example

```
CubeCreate('y2ksales', 'Actvsbud', 'Region', 'Model','Account1', 'Month');
```

This example creates a cube named y2ksales using the dimensions Actvsbud, Region, Model, Account1, and Month.

CubeDestroy

This function deletes a specified TM1 cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CubeDestroy(Cube);
```

Argument	Description
Cube	The name of the cube you want to delete.

Example

```
CubeDestroy('y2ksales');
```

This example deletes the cube named y2ksales.

CubeExists

Use CubeExists to determine if a specific cube exists on the server from which a TurboIntegrator process is executed.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The function returns 1 if the cube exists on the server, otherwise it returns 0.

Syntax

```
CubeExists(CubeName);
```

Argument	Description
CubeName	The name of the cube whose existence you want to confirm.

Example

```
CubeExists('Inventory');
```

This example determines if the Inventory cube exists on the server.

CubeGetLogChanges

This function returns the Boolean value of the Logging property for a specified cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The Logging property is set in the TM1 Security Assignments dialog box and stored in the }CubeProperties control cube. If Logging is turned on for a cube, the function returns 1. If logging is turned off the function returns 0.

Syntax

```
CubeGetLogChanges(CubeName);
```

Argument	Description
CubeName	The cube for which you want to return the value of the Logging property.

Example

```
CubeGetLogChanges('2002sales');
```

If Logging is turned on for the 2002sales cube, the function returns 1.

CubeSaveData

CubeSaveData() serializes a cube.

Syntax

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

To improve performance, transaction logging may be disabled while loading data. To safeguard newly loaded data in the unlikely event of a server crash, the changes can be serialized to disk. SaveDataAll has been used to serialize data to disk and to truncate the transaction log. When processing a SaveDataAll command, the server acquires a READ lock on every cube and an IX lock on every changed cube. This can cause significant contention with user activity if SaveDataAll is run during periods of user activity.

Typically not all the cubes affected by SaveDataAll need to be serialized since not all cubes are typically loaded with new data. CubeSaveData is used to serialize an individual cube to disk. CubeSaveData serializes the cube's data that has been committed to memory including the modifications that have been performed against it in the current TurboIntegrator process but not yet committed.

```
CubeSaveData(Cube);
```

Argument	Description
Cube	The name of the cube you want to serialize.

Example

```
CubeSaveData ('SalesCube');
```

Consider the following TurboIntegrator process code:

```
CellPutN(500, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');

CubeSaveData('y2ksales');

CellPutN(1000, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');
```

When the CubeSaveData command is processed, the value of 500 for the January Sales cell will be included in the cube's serialization to disk, even though it has not yet been committed. The update of the January Sales cell to 1000 will not be part of the serialization.

Transaction Log

A new transaction entry appears in the Transaction log when CubeSaveData has been run. When processing a transaction log file during recovery, all updates to a cube that have been applied so far will be discarded when a CubeSaveData directive against the cube is encountered as all of the updates have already been serialized to the cube.

Server Crash Recovery

The SaveDataAll command takes advantage of the fact that all cubes are locked during its processing and truncates the transaction log knowing that all updates performed before serialization have been safely stored to disk. This is not the case for CubeSaveData so you must modify the way data recovery is performed when a cube has been serialized.

The transaction log file could contain records that represent changes that are older than the most recent data in the cube and should not be applied when data is being recovered.

CubeSetConnParams

This function is used to encrypt the password for a virtual cube in the }CubeProperties cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CubeSetConnParams(cubeName, providerName, dataSourceLocation,dataSourceName,
dataSourceCatalog, userID, password,sapClientLang,providerString);
```

Argument	Description
cubeName	The name of the cube for which you want to set the password.
providerName	
dataSourceLocation	Name your administrator assigns to a set of catalogs at a particular location. In Microsoft Analysis Services, this is the name of a registered server.
dataSourceName	
dataSource catalog	The name assigned by your administrator to a particular collection of databases (Cubes, Dimensions and other objects). For MAS, this is the name of the database.
UserID	A valid username for the database.
Password	Password to use for this data source.

Argument	Description
sapClientLang	SAP language setting.
providerString	

Example

```
CubeSetConnParams(sc, TM1OLAP, tm1server, , sdata, admin,apple,, ,);
```

CubeSetLogChanges

This function sets the LOGGING property for a cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CubeSetLogChanges(Cube, LogChanges);
```

Argument	Description
Cube	The name of the cube for which you want to set the LOGGING property.
LogChanges	The Boolean value you want to assign to the property. 1= LOGGING on, 0 = LOGGING off.

CubeUnload

This function unloads a specified cube, along with all associated cube views, from memory.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CubeUnload(CubeName);
```

Argument	Description
CubeName	The cube you want to unload from memory.

Example

```
CubeUnload('ManufacturingBudget');
```

This example unloads the ManufacturingBudget cube, and any associated views, from server memory.

Data Reservation TurboIntegrator Functions

Use the following TurboIntegrator functions to programmatically obtain, release and manage Data Reservations.

For more details about using the Data Reservation feature, see "Using Data Reservations" in the IBM Cognos TM1 *Developer Guide*.

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.
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.

Argument	Description
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.
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.

Date and Time TurboIntegrator Functions

These functions format and parse dates and times in a wide variety of formats and locales.

FormatDate

The FormatDate function formats a date value according to a formatter defined with the NewDateFormatter function.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
FormatDate(Date, <Pattern>, <Index>)
```

Arguments

Argument	Description
Date	A date value. The type of value, serial or UNIX, should correspond to the formatter used.
Pattern	Pattern used for formatting dates. Refer to http://userguide.icu-project.org/formatparse/datetime for a complete list of format syntax. If an empty string is passed, then the format is determined by the locale based on the FormatterStyle and FormatterType parameters that were used with the NewDateFormatter function.
Index	Index returned by a call to the NewDateFormatter function. The default value is 0. If no date formatter exists at the index, then a default formatter is used as though it had been created with the following call: `NewDateFormatter('', 'Etc/UTC', 'serial', 'medium', 'date')`

Example

```
sDate = FormatDate(18000);
```

NewDateFormatter

The NewDateFormatter function defines a date formatter. It returns an index for use in the ParseDate and FormatDate functions. The indices start at 0 and go up by one for each call to NewDateFormat. Date formatters are valid during execution of the process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
NewDateFormatter(Locale, <TimeZone>, <UseUNIXTime>, <FormatterStyle>,
<FormatterType>, <TimeType>)
```

Arguments

Argument	Description
Locale	Locale used for parsing or formatting dates. If an empty string is passed, then the operating system locale is used. Locales are specified in the format language[_territory][.variant]. For example, `cs_CK` is the Czech language and Czech Republic.
TimeZone	Timezone used for parsing or formatting dates. Refer to http://en.wikipedia.org/wiki/List_of_tz_database_time_zones for a complete list of time zones. If not specified, the time zone used is `UTC` (`'Etc/UTC'`).
UseUNIXTime	If `'unix'` is specified, then times are treated as milliseconds since January 1, 1970. Otherwise, they are treated in TM1 serial format. Note that only dates later than January 1, 1970 can be processed even if TM1 serial format is used.
FormatterStyle	Controls the date format used when an empty pattern is specified to the FormatDate or ParseDate functions. Valid values are `'full'`, `'long'`, `'medium'` or `'short'`. The default is `'medium'`.
FormatterType	Controls the type of format used when an empty pattern is specified to the FormatDate or ParseDate functions. Valid values are `'time'`, `'date'` or `'datetime'`. The default is `'date'`.

Example

```
dfUNIX = NewDateFormatter('', 'Etc/UTC', 'unix');

dfStyleFullDateTime = NewDateFormatter('en_us', 'America/Toronto',
'serial', 'full', 'datetime');
```

ParseDate

The ParseDate function parses a date string according to a formatter defined with the NewDateFormatter function.

A date value that is either serial or UNIX, depending on the formatter specified, is returned. If the date cannot be parsed then an undefined value is returned. This can be tested with the ISUND function.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ParseDate (DateString, <Pattern>, <Index>)
```

Arguments

Argument	Description
DateString	A date string.
Pattern	Pattern used for parsing dates. Refer to http://userguide.icu-project.org/formatparse/datetime for a complete list of format syntax. If an empty string is passed, then the format is determined by the locale based on the FormatterStyle and FormatterType parameters that were used with the NewDateFormatter function.
Index	Index returned by a call to the NewDateFormatter function. The default value is 0. If no date formatter exists at the index, then a default formatter is used as though it had been created with the following call: `NewDateFormatter('', 'Etc/UTC', 'serial', 'medium', 'date')`

Example

```
nDate = ParseDate('2011/11/24', 'yyyy/MM/dd');
```

Dimension Manipulation TurboIntegrator Functions

These functions facilitate the manipulation of dimensions.

DimensionCreate

DimensionCreate creates a new dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
DimensionCreate(DimName);
```

Argument	Description
DimName	The name you want to assign to the dimension.

Example

```
DimensionCreate('Product');
```

This example creates the Product dimension.

DimensionDeleteAllElements

This function deletes all the elements in a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Note: Deleting an element deletes all cube data identified by that element. However, if you use DimensionDeleteAllElements to delete elements, then recreate those elements with the same names in the Metadata tab, any data points in a cube identified by the elements will be retained after rebuilding the dimension.

This function is useful for recreating dimension hierarchies.

Syntax

```
DimensionDeleteAllElements(DimName);
```

Argument	Description
DimName	The name of the dimension from which you want to delete all elements.

Example

```
DimensionDeleteAllElements('Model');
```

This example deletes all elements in the Model dimension.

DimensionDestroy

This function deletes a dimension from the TM1 database.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
DimensionDestroy(DimName);
```

Argument	Description
DimName	The name of the dimension you want to delete.

Example

```
DimensionDestroy('Product');
```

This example deletes the Product dimension from the TM1 database.

DimensionElementComponentAdd

Adds a component (child) to a consolidated element. Note that you cannot use this function in the Epilog procedure of a TurboIntegrator process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
DimensionElementComponentAdd(DimName, ConsolidatedElName,ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
ConsolidatedElName	The element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
DimensionElementComponentAdd('Measures', 'Net Sales', 'Expenses', -1);
```

This example adds the child Expenses to the Net Sales consolidation in the Measures dimension. The child has a weight of -1 in the consolidation.

DimensionElementComponentAddDirect

DimensionElementComponentAddDirect adds a component (child) to a consolidated element by directly editing a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The default means of editing a dimension in Cognos TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementComponentAdd) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementComponentAddDirect(DimName, ConsolidatedElName,ElName, ElWeight);
```

Argument	Description
DimName	The parent dimension of the consolidated element to which you want to add a child.
ConsolidatedElName	The consolidated element to which you want to add a child.
ElName	The name of the child element.
ElWeight	The weight of the child element.

Example

```
DimensionElementComponentAddDirect('Measures', 'Net Sales', 'Expenses',
-1);
```

This example adds the child Expenses to the Net Sales consolidation in the Measures dimension. The child has a weight of -1 in the consolidation.

DimensionElementComponentDelete

This function deletes a component (child) from a consolidated element.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
DimensionElementComponentDelete(DimName, ConsolidatedElName,ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
DimensionElementComponentDelete('Region', 'Benelux','Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Region dimension.

DimensionElementComponentDeleteDirect

This function deletes a component (child) from a consolidated element by directly editing the dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The default means of editing a dimension in Cognos TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementComponentDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementComponentDeleteDirect(DimName, ConsolidatedElName,ElName);
```

Argument	Description
DimName	The parent dimension of the consolidated element from which you want to delete a child.
ConsolidatedElName	The consolidated element from which you want to delete a child.
ElName	The name of the child element you want to delete.

Example

```
DimensionElementComponentDeleteDirect('Region', 'Benelux','Belgium');
```

This example deletes the Belgium child from the Benelux consolidation in the Region dimension.

DimensionElementDelete

This function deletes an element from a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Note: Deleting an element deletes all cube data identified by that element.

Syntax

```
DimensionElementDelete(DimName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
ElName	The element you want to delete.

Example

```
DimensionElementDelete('Region', 'Belgium');
```

This example deletes the element Belgium from the Region dimension.

DimensionElementDeleteDirect

This function deletes an element from a dimension by directly editing the dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Note: Deleting an element deletes all cube data identified by that element.

The default means of editing a dimension in Cognos TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementDelete) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementDeleteDirect(DimName, ElName);
```

Argument	Description
DimName	The dimension that contains the element you want to delete.
ElName	The element you want to delete.

Example

```
DimensionElementDeleteDirect('Region', 'Belgium');
```

This example deletes the element Belgium from the Region dimension.

DimensionElementInsert

This function adds an element to a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. You can use this function to add numeric, string, or consolidated elements. Note that you cannot use this function in the Data or Epilog procedures of a TurboIntegrator process.

Syntax

```
DimensionElementInsert(DimName, InsertionPoint, ElName,ElType);
```

Argument	Description
DimName	The dimension to which you want to add a new element.
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is empty, the new element is added to the end of the dimension.
ElName	The name you want to assign to the new element.
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
DimensionElementInsert('Region', 'Belgium', 'Netherlands','N');
```

This example adds the numeric element Netherlands to the Region dimension. Netherland displays immediately before Belgium in the dimension definition.

DimensionElementInsertDirect

This function adds an element to a dimension by directly editing the dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. You can use this function to add numeric, string, or consolidated elements.

The default means of editing a dimension in TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionElementInsert) are used in the metadata tab of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element insertion needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionElementInsertDirect(DimName, InsertionPoint, ElName,ElType);
```

Argument	Description
DimName	The dimension to which you want to add a new element.
InsertionPoint	An existing dimension element. The element being added to the dimension will be inserted immediately before this existing element. If this parameter is empty, the new element is added to the end of the dimension. Note that this function is optimized for the case where the InsertionPoint is passed as an empty string.
ElName	The name you want to assign to the new element.

Argument	Description
ElType	The element type. There are three possible ElType values: N - Signifies a numeric element. S - Signifies a string element. C - Signifies a consolidated element.

Example

```
DimensionElementInsertDirect('Region', 'Belgium', 'Netherlands','N');
```

This example adds the numeric element Netherlands to the Region dimension. Netherlands displays immediately before Belgium in the dimension definition.

DimensionElementPrincipalName

This function returns the principal name of an element or element alias.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

TurboIntegrator must use principal element names when updating dimensions; element aliases cannot be used. This function is therefore useful for determining principal element names while attempting to update a dimension when only element aliases are available to the TurboIntegrator process.

Syntax

```
DimensionElementPrincipalName( DimName, ElName )
```

Argument	Description
DimName	The name of the dimension from which you want to retrieve a principal element name.
ElName	An element name. ElName can be either an element alias or a principal element name.

Example

If ElName is not in the currently saved version of DimName, the function returns ElName.

If ElName is in DimName, whether as an element alias or a principal element name, it returns the principal name of the element.

DimensionExists

Use DimensionExists to determine if a specific dimension exists on the server from which a TurboIntegrator process is executed.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. The function returns 1 if the dimension exists on the server, otherwise it returns 0.

Syntax

```
DimensionExists(DimName);
```

Argument	Description
DimName	The name of the dimension whose existence you want to confirm.

Example

```
DimensionExists('Region');
```

This example determines if the Region dimension exists on the server.

DimensionSortOrder

This function sets a sort type and sense for dimension elements and for components of consolidated elements within a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. The sort order defined byDimensionSortOrder determines how the subset All dipsplays in the Subset Editor.

DimensionSortOrder sets properties for a dimension; the dimension is not actually sorted until it is saved on the server.

Syntax

```
DimensionSortOrder(DimName, CompSortType, CompSortSense,ElSortType , ElSortSense);
```

Argument	Description
DimName	The name of the dimension for which you want to set a sort order.
CompSortType	Defines how components of consolidated elements appear in the dimension. There are two CompSortType values: ByInput - Retains the order in which components were originally inserted into consolidations. ByName - Sorts components of consolidations by name.

Argument	Description
CompSortSense	Defines the sort sense for components of consolidations. This is a required argument, but it applies only when the CompSortType is ByName. There are two possible CompSortSense values: Ascending - Sorts components of consolidations in ascending alphabetical order. Descending - Sorts components of consolidations in descending alphabetical order.
ElSortType	Defines a sort order for dimension elements. There are four possible ElSortType values: ByInput - Retains the order in which elements were originally inserted into the dimension. ByName - Sorts dimension elements by name. ByLevel - Sorts dimension elements by level. ByHierarchy - Sorts dimension elements by hierarchy.
ElSortSense	Defines the sort sense for dimension elements. This is a required argument, but it applies only when the ElSortType is ByName or ByLevel. There are two possible ElSortSense values: Ascending - Sorts dimension elements in ascending order, either alphabetically or by level. Descending - Sorts dimension elements in descending order, either alphabetically or by level.

Example

```
DimensionSortOrder ('Region', 'ByName', 'Descending','ByLevel', 'Ascending');
```

This example sets a sort order for the Region dimension. All dimension elements are sorted in ascending by level, and any components of consolidations are sorted in descending alphabetical order.

DimensionUpdateDirect

This function performs a full rewrite of a dimension that has been subject to direct editing in a TurboIntegrator process, essentially compacting the memory footprint of the dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

A dimension that undergoes a series of direct-only edits (element deletions, in particular) will eventually use more memory than its fully-rewritten counterpart would. This function can optionally be used after directly editing a dimension with DimensionElementInsertDirect, DimensionElementDeleteDirect, DimensionElementComponentAddDirect, DimensionElementComponentDeleteDirect, and/or DimensionTopElementInsertDirect. Calling DimensionUpdateDirect incurs an initial full-copy memory cost, however it can be used to guarantee that the dimension is at its smallest possible memory footprint after processing is complete.

Syntax

```
DimensionUpdateDirect(DimName);
```

Argument	Description
DimName	The name of the dimension you want to rewrite.

Example

```
DimensionUpdateDirect('Region');
```

This example rewrites the Region dimension.

DimensionTopElementInsert

This function creates a root element in a dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. If the dimension already has a single root, then this element will not be created.

Syntax

```
DimensionTopElementInsert(DimName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
DimensionTopElementInsert('Region', 'Netherlands', 'World');
```

This example adds the root element World to the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

DimensionTopElementInsertDirect

This function creates a root element in a dimension by directlt editing the dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. If the dimension already has a single root, then this element will not be created.

The default means of editing a dimension in Cognos TM1 is to use a whole-copy editing pattern. In that pattern, an editing copy of the dimension is created, edits are applied to the editing copy, then finally the actual dimension is rewritten using the editing copy as a template. TurboIntegrator supports whole-copy editing automatically whenever dimension editing TurboIntegrator functions (like DimensionTopElementInsert) are used in the Metadata procedure of the process. TurboIntegrator automatically creates the editing copy and applies editing operations to it, then rewrites the actual dimension at the end of the Metadata procedure.

Direct edits are different in that no editing copy is involved. Instead, the operations are performed directly on the actual dimension. There are two different, specialized use cases for which this type of direct editing is intended:

- When the purpose of the TurboIntegrator process is to make a small change to a large dimension. In this case, direct editing will be more efficient because it avoids copying and completely rewriting the large dimension.
- When the purpose of the TurboIntegrator process is to load large volumes of data into a cube. In this case the process' Metadata procedure is deliberately kept empty, and any element modification needed to support data loading is performed using direct calls in the Data procedure. When the Metadata procedure is empty, the process skips an entire iteration over the external datasource, which can result in faster data loads.

Syntax

```
DimensionTopElementInsertDirect(DimName, InsertionPoint, ElName);
```

Argument	Description
DimName	The dimension for which you want to create a root element.
InsertionPoint	An existing dimension element. The root element being added to the dimension will be inserted immediately before this existing element.
ElName	The name you want to assign to the new root element.

Example

```
DimensionTopElementInsertDirect('Region', 'Netherlands', 'World');
```

This example adds the root element World to the Region dimension. World is inserted displays immediately before Netherlands in the dimension definition.

ODBC TurboIntegrator Functions

These functions facilitate the ODBC manipulation.

ODBCClose

This function closes a connection to an ODBC data source.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ODBCClose(Source);
```

Argument	Description
Source	The name of an open ODBC data source.

Example

```
ODBCClose('Accounting');
```

This example closes the connection to the Accounting ODBC source.

ODBCOpen

This function opens an ODBC data source for output.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ODBCOpen(Source, ClientName, Password);
```

Argument	Description
Source	An ODBC data source name.
ClientName	A valid client on the data source.
Password	A password for the ClientName.

Example

```
ODBCOpen('Accounting', 'Jdoe', 'Bstone');
```

This example opens the Accounting ODBC data source for the Jdoe client using the password Bstone.

ODBCOPENEx

This function opens an ODBC data source for output specifying that the connection should be opened as a Unicode connection.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Format is: ODBCOPENEx (dataset name, dataset client name, client password, (use-Unicode-interface flag))

Syntax

```
ODBCOpenEx(Source, ClientName, Password, UseUnicodeODBC);
```

Argument	Description
Source	An ODBC data source name.
ClientName	A valid client on the data source.
Password	A password for the ClientName.
UseUnicodeODBC	Defines the type of Unicode connection to use.

Example

```
ODBCOpenEx( TestTable, sa, , 1 );
```

```
chinese= ;
chinese = CHARW( 37123 );
fieldval = chinese | SomeNewText;
sql= Update TestTable set ForeName = N | fieldval |  WHERE CustomerId= 1
ODBCOUTPUT( Unicode, sql );
```

The result SQL statement looks like:

```
Update TestTable set ForeName = N?SomeNewText WHERE
CustomerId = 1
```

ODBCOutput

This function executes an SQL update query against an open ODBC data source.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. You should use the ODBCOpen function to open the data source before calling ODBCOutput, and use ODBCClose to close the data source before exiting the process.

Syntax

```
ODBCOutput(Source, SQLQuery, [SQLQuery2, SQLQuery3, ...]);
```

Argument	Description
Source	The ODBC data source against which you want to run a query.

Argument	Description
SQLQuery	An SQL query statement. Though ODBCOutput was developed to update tables, it can be used to execute any SQL query on the data source. In circumstances where the SQL query statement exceeds 255 characters, you should split the query into multiple SQLQuery arguments (SQLQuery2, SQLQuery3, etc.). This lets you create query statements that exceed the 255 character limit for TurboIntegrator arguments. When the ODBCOutput function is executed, all SQLQuery arguments are concatenated and the query is successfully executed.

Example

```
ODBCOutput('Accounting', 'INSERT [CategoryID], [CategoryName]FROM Categories;');
```

This example executes the specified query against the Accounting data source.

SetODBCUnicodeInterface

This function sets whether the ODBC interface should use the Unicode "wide" functions or the regular single-byte character functions.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Setting this function to 1 uses the wide character ODBC interface.

Some ODBC driver support either the older single-byte interface as well as a Unicode style 'wide-character' interface, where characters are passed and retrieved as 16-bit quantities. If the driver chosen does not support one or the other style, a flag is provided to force TurboIntegrator to use a particular style of interface.

Syntax

```
SetODBCUnicodeInterface=1
```

Argument	Description
1	Use the wide character ODBC interface.
0	Use the single-byte interface.

Process Control TurboIntegrator Functions

These functions pertain to process control.

ProcessError

This function causes an immediate termination of a process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Processes terminated with this function are flagged with an error status.

Syntax

```
ProcessError;
```

Arguments

None.

ExecuteCommand

This function executes a command line during a process. You can use ExecuteCommand to run a desktop application, but not a service.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

If you use ExecuteCommand to run an executable, the following conditions apply:

- If the CommandLine argument specifies only the name of a file to be executed, a Windows server looks for the file in both the server database directory and in the directory where Tm1s.exe resides. A UNIX server looks for the file only in the server database directory.
- If the CommandLine argument uses a relative path prefix, both the Windows and UNIX server attempt to locate the file in the server database directory only.
- On either the Microsoft Windows or UNIX server, you can pass an absolute path to the CommandLine argument to execute a file in any location..

Syntax

```
ExecuteCommand(CommandLine, Wait);
```

Argument	Description
CommandLine	The command line you want to execute.
Wait	Indicates if the process should wait for the command to complete execution before continuing to the next process statement. An argument value of 0 causes the process to proceed to the next statement without waiting for the command line to execute. An argument value of 1 causes the process to wait for the command line to successfully execute before proceeding to the next statement.

ExecuteProcess

This function lets you execute a TurboIntegrator process from within another process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ExecuteProcess(ProcessName, [ParamName1, ParamValue1,ParamName2, ParamValue2]);
```

Argument	Description
ProcessName	The name of the process to be executed. This process must reside on the same server as the process from which ExecuteProcess is called. If the process named by this argument cannot be found at runtime, the calling process is immediately terminated. (TurboIntegrator does not check for a valid ProcessName at compilation.)
ParamName	The name of an existing parameter of the process to be executed. This argument is required only if the process to be executed uses parameters.
ParamValue	A valid value for the ParamName parameter. If you specify a ParamName argument, you must specify a corresponding ParamValue. The ParamName and ParamValue arguments must occur in ordered pairs, with the name of the parameter followed by the value. You must specify a ParamName and corresponding ParamValue for each parameter of the process to be executed.

The parameter names passed in the ExecuteProcess function are matched at runtime against the parameter names specified in the process to be executed. If the passed names cannot be found in the parameter list of the process to be executed, a serious error results, causing the immediate termination of the process from which ExecuteProcess is called.

Return Values

ExecuteProcess returns a real value that can be tested against one of the following return value functions:

Function	Description
ProcessExitByChoreQuit()	indicates that the process exited due to execution of the ChoreQuit function
ProcessExitNormal()	indicates that the process executed normally
ProcessExitMinorError()	indicates that the process executed successfully but encountered minor errors
ProcessExitByQuit()	indicates that the process exited because of an explicit "quit" command
ProcessExitWithMessage()	indicates that the process exited normally, with a message written to `tm1server.log`
ProcessExitSeriousError()	indicates that the process exited because of a serious error

Function	Description
ProcessExitOnInit()	indicates that the process aborted during initialization
ProcessExitByBreak()	indicates that the process exited because it encountered a ProcessBreak function

Example

To record when a process called by ExecuteProcess fails because of a serious error, use code similar to the following:

```
return_value = ExecuteProcess('create_sales_cube');
ASCIIOutput('C:\temp\process_return_value.txt', 'Process exited
with serious errors at', TIME, 'on', TODAY);if(return_value = ProcessExitSeriousError() )
endif;
```

GetProcessErrorFileDirectory

This function returns the full pathname, with trailing slash, of the directory where TurboIntegrator process error files are written.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. By default, all process error log files are written to the data directory of the server on which the process resides.

Syntax

```
GetProcessErrorFileDirectory;
```

Arguments

None.

GetProcessErrorFilename

This function returns the name of the TurboIntegrator process error log file associated with a process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. If the process has not yet generated an error log file, the function returns an empty (null) string.

Important: A process error log file is not generated until all statements in a given process tab (Prolog, Metadata, Data, or Epilog) have executed. Accordingly, you can use GetProcessErrorFilename to check if any previous tabs have generated an error log file, but you cannot use the function to determine if the current process tab causes errors to be written to a log file.

For example, by determining that GetProcessErrorFilename returns a non-null string in the Epilog tab, you can tell that errors were generated in the Prolog, Metadata, or Data tabs. However, you cannot use GetProcessErrorFilename in the Data tab to determine if the Data tab generates errors.

Syntax

```
GetProcessErrorFilename;
```

Arguments

None.

GetProcessName

This function returns as a string the name of the current process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
GetProcessName()
```

Arguments

None.

```
Name = GetProcessName();
```

If

The If statement allows a process to execute a statement or series of statements when a given expression is true.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. You can use arithmetic operators, logical operators, and comparison operators to construct an expression.

The TurboIntegrator If statement differs from the Rules IF function in that the TurboIntegrator statement can accept multiple ElseIf or Else statements to evaluate multiple expressions, while the Rules IF function can evaluate only one expression.

You can nest up to 20 If/ElseIf/Else statements in a TurboIntegrator process. If you exceed 20 nested If/ElseIf/Else statements, you will receive an error when attempting to save the process.

Syntax

```
If(expression);
statement1;
ElseIf(expression);
statement2;
ElseIf(expression);
statement3;
Else;
statement4;
EndIf;
```

Arguments

None.

Examples

```
If (x=5);
 ASCIIOutput('c:\temp\if.txt','x equals five');
ElseIf (x=1);
 ASCIIOutput ('c:\temp\if.txt', 'x equals one');
ElseIf (x=2);
 ASCIIOutput ('c:\temp\if.txt', 'x equals two');
ElseIf (x=3);
 ASCIIOutput ('c:\temp\if.txt', 'x equals three');
```

```
ElseIf (x=4);
 ASCIIOutput ('c:\temp\if.txt', 'x equals four');
Else;
 ASCIIOutput ('c:\temp\if.txt', 'x falls outside expected range');
EndIf;
```

This example evaluates the value of X. If X=5, the ASCIIOutput function is executed to write the string "x equals five" to c:\temp\if.txt. If X does not equal 5, the first ElseIf statement is evaluated. If X=1, the ASCIIOutput function is executed to write the string "x equals one" to c:\temp\if.txt. This processing continues until the EndIf is executed.

Simple If statements can also be constructed without the use of ElseIf, as in this example:

```
IF(expression);
 statement1;
ELSE;
 statement2;
ENDIF;
```

ItemReject

This function rejects a source record and places it in the error log, along with a specified error message.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ItemReject(ErrorString);
```

Argument	Description
ErrorString	The error message you want written to the error log when a record is rejected.

Example

```
ItemReject(' Value outside of acceptable range.');
```

This example places a source record in the error log, along with the error message 'Value outside of acceptable range.' when the source record contains a value that is beyond a defined range.

ItemSkip

This forces a process to skip the current data source item.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ItemSkip;
```

Arguments

None.

ProcessBreak

This function stops processing source data and proceeds to the Epilog portion of a process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ProcessBreak;
```

Arguments

None.

ProcessError

This function causes an immediate termination of a process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Processes terminated with this function are flagged with an error status.

Syntax

```
ProcessError;
```

Arguments

None.

ProcessQuit

This function terminates a TurboIntegrator process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ProcessQuit;
```

Arguments

None.

While

The TurboIntegrator While statement allows a process to repeat a series of statements while a given condition is true.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

While statements can be nested.

Syntax

```
WHILE(logical expression);

statement1;

statement2;
```

```
...

statement n;

END;
```

Note: All WHILE statements must conclude with an END statement.

Arguments

None.

Synchronized

Synchronized()is used in a TurboIntegrator script to force serial execution of a designated set of TurboIntegrator processes.

The synchronized() function uses the following syntax.

```
synchronized(string)
```

Parameters

synchronized() takes a single required parameter that is a user-defined name for a lock object. This lock object name can be used in multiple TurboIntegrator processes in order to serialize their execution as a group.

Parameter	Description Value/Required/Default
`lockName`	The user-defined name of a lock object on which to synchronize. Names are case-insensitive and embedded spaces are ignored. Names may not exceed 1023 characters in length. String/Yes/None

Semantics

A TurboIntegrator process may make any number of calls to synchronized(), with any number of lock objects. Serializing is effective from the time synchronized() is called, until the containing transaction completes.

For example, if synchronized() is called from a subprocess (Ps) of master process (Pm) or master chore (Cm), the Lock Object is "released" when Pm or Cm completes. The exception is that a SaveDataAll (SDA) prematurely "ends" a transaction mid-process execution; this applies to Lock Objects as well.

The synchronized() call may be placed anywhere within a TurboIntegrator script, but serialization applies to the entire TurboIntegrator process when it is encountered.

Consider a TurboIntegrator process with a synchronized() call somewhere in the "middle" of its script, and an operation O1 preceding that call. Two instances of this TurboIntegrator process may start at the same time. It is possible for one instance to run to completion, including its call to synchronized(), before the second instance reaches its synchronized() call. In this case, the two processes appear to the user to have run concurrently. If, instead, the second process does reach its synchronized() call before the first completes, it will undo any work it had done (O1) and wait for the first to complete. In this case, the two processes appear to the user to have serialized.

To avoid such confusion, and to optimize the use of synchronized(), it is recommended (but not enforced) that synchronized() calls be the first statements of a TurboIntegrator process.

Example

Consider that TurboIntegrator process P needs to update two cubes, Cube_1 and Cube_2.

Other TurboIntegrator processes may also need to update Cube_1 or Cube_2.

To cause all TurboIntegrator processes that will update Cube_1 or Cube_2, to run one at a time, P could call synchronized() in the following way:

```
sCube_1='Cube_1';
sCube_2='Cube_2';
sE1='Elm1';
sE2='Elm2';
sE4='Units';
sE5='Price';

Synchronized( sCube_1 );
Synchronized( sCube_2 );

CellPutn( 111, sCube_1, sE1, sE2 );
CellPutn( 9.99, sCube_2, sE4, sE5 );

# ...
```

Other TurboIntegrator processes that will update Cube_1 or Cube_2 must also call `synchronized( sCube_1 )` and/or `synchronized( sCube_2 )` in a similar way.

In this example, the two lock objects' names were chosen to be the same as the cubes' names. But a lock object's name does not have to be the same as other Cognos TM1 objects (cubes, dimensions, subsets, etc.).

Rules Management TurboIntegrator Functions

These functions facilitate rules management.

CubeProcessFeeders

This function reprocesses all feeders in the rules for a specified cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function reprocesses all feeders in the rules for a specified cube. The feeders are normally reprocess automatically when a rule file edit is saved, however, if the data changes, and those data changes will change some conditional feeders, this function will need to be called to get those conditional feeders re-evaluated.

Syntax

```
CubeProcessFeeders(CubeName);
```

Argument	Description
CubeName	The cube for which you want to reprocess feeders.

Example

```
CubeProcessFeeders('2003sales');
```

This example reprocesses all feeders in the rules for the 2003sales cube.

CubeRuleAppend

This function appends a single line of rule text to a TM1 cube rule.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Essentially, this function adds a single line of text to a rule (.rux) file. The line of text is typically a rule statement, but can also be a comment. If there is no rule associated with the cube at the time this function is executed, a new rule will be created, containing only the passed line.

Syntax

```
CubeRuleAppend(CubeName, RuleText, IsCalculationRule);
```

Argument	Description
CubeName	The name of the cube associated with the rule to which you want to append a line of text.

RuleText	The single line of text you want to append to the rule.
	The entire line of text you add must be enclosed in single quotes and must adhere to rules syntax conventions.
	If the line of text includes any element references, the element names must be enclosed in double single quotes to escape the single quotes that normally enclose element names. For example, a reference to an element named CL3 must be specified as `[''CL3'']`.
	The following are examples of valid lines of text you might append to a rule:
	`'[''CL3''] = [''CL4''] + [''Trial''];'`
	`'skipcheck;'`
	`'[''Trial''] => [''CL3''];'`

Argument	Description
IsCalculationRule	The IsCalculationRule parameter declares whether the line should be inserted just above any feeder section that might exist in the cube rule. If the IsCalculationRule parameter is omitted, or passed as 0.0, then the new line will simply be appended to the end of the rule. Because rule (.rux) files consist of a calculation section followed by an optional feeder section, any appended lines that are calculation rule statements (or corresponding comments) should use a 1.0 for this argument to ensure that the new line is inserted in at the appropriate location in the rule file.

Examples

```
CubeRuleAppend( 'MyCube', '[''CL3''] = [''CL4''] + [''Trial''];', 1.0 );
```

This example inserts the calculation statement `['CL3'] = ['CL4'] + ['Trial'];` at the end of the calculation section of the rule for the MyCube cube.

```
CubeRuleAppend( 'MyCube', '[''Trial''] => [''CL3''];', 0.0 );
```

This example inserts the feeder statement `['Trial'] => ['CL3'];` at the end of the rule for the MyCube cube.

CubeRuleDestroy

This functions deletes any rule that exists for a specified cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
CubeRuleDestroy(CubeName);
```

Argument	Description
CubeName	The name of the cube associated with the rule that you want to delete

Example

```
CubeRuleDestroy('SalesProjections');
```

This example deletes the rule for the SalesProjectionscube.

DeleteAllPersistentFeeders

Deletes any .feeder files that have persisted.

When this function is used, all cubes are marked as "do not save feeders" so a subsequent "SaveData" will not persist feeders which means all feeders will be re-calculated on a server re-start.

Syntax

```
DeleteAllPersistentFeeders;
```

Argument	Description
none	There are no arguments for this function.

ForceSkipCheck

Forces the query to perform as if the cube had a SKIPCHECK in the rules.

This means that the query will process only values actually in the cube, as opposed to (the no SKIPCHECK case) where every possible cell would be enumerated looking for values. This function sets the state of the view query to select only values in the cube. The function must be added to the Prolog section of the Turbo Integrator process. By placing the ForceSkipCheck() in the Prolog it effects the entire view query of data elements to follow.

Syntax

```
ForceSkipCheck()
```

Argument	Description
none	There are no arguments for this function.

RuleLoadFromFile

This function creates a TM1 rule for a specified cube from a text file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The text file must be formatted according to TM1 rules conventions. Each rule statement must conclude with a semi-colon (;) and comments must be prefixed with the # character.

If a rule already exists for the specified cube, the existing rule is overwritten by the rule created by RuleLoadFromFile.

Syntax

```
RuleLoadFromFile(Cube, TextFile);
```

Argument	Description
Cube	The name of the cube for which you want to create a rule.

Argument	Description
TextFile	The name of the text file from which you want to create a rule. You can specify the full path to this file, including file name and extension. (Example 1 below.) If you specify only the file name and extension, TurboIntegrator looks for the file in the server's data directory. If you do not specify a file extension, TurboIntegrator assumes the .rux extension by default. (Example 2 below.)

If you leave the TextFile argument empty, TurboIntegrator looks for a source file with the same name as the cube (but with a .rux extension) in the server's data directory. (Example 3 below.)

Example

```
RuleLoadFromFile('Sales', 'C:\temp\cuberule.txt');
```

This example uses the contents of the cuberule.txt file in the C:\temp directory to create a rule for the Sales cube.

```
RuleLoadFromFile('Sales', 'cuberule');
```

This example creates a rule for the Sales cube using the file named cuberule.rux in the server's data directory.

```
RuleLoadFromFile('Sales', ' ');
```

This example creates a rule for the Sales cube using the file named Sales.rux in the server's data directory.

Sandbox Functions

These functions are used with sandboxes.

GetUseActiveSandboxProperty

This function returns a Boolean value that indicates whether a process reads and writes data to the base data or to the user's active sandbox.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The default is for processes to read and write to the base data.

- If the return is 0, the process is currently reading and writing to the base data.
- If the return is 1, the process is currently reading and writing to the active sandbox.

Note: This function returns the permanent value for this property as set in the Architect / Server Explorer user interface *unless*you have used the

SetUseActiveSandboxProperty function in the process. In that case, the value for this property is determined by the value that was last set with the SetUseActiveSandboxProperty function.

Syntax

```
GetUseActiveSandboxProperty()
```

None.

Example

```
return_value = GetUseActiveSandboxProperty();
```

This example will return a Boolean value indicating whether the process is currently reading and writing cube data to the active sandbox or to the base data.

ServerSandboxesDelete

ServerSandboxesDelete allows administrators to discard user sandboxes that match certain criteria.

This is a TM1 TurboIntegrator function, valid in TurboIntegrator processes.

Sandboxes are private workspaces in which a user can enter and store data values separate from IBM Cognos TM1 base data. Sandboxes are stored on disk and, when in use, in memory. This functionality operates server side and is available through TurboIntegrator and the API function ServerSandboxesDelete. Using this feature in a TurboIntegrator process, administrators can schedule maintenance using automated chores.

Description

This function uses a "predicate" to describe the sandbox being deleted. A predicate can be read as: "Delete sandboxes whose *attribute* is *condition value*."

For example: "Delete sandboxes whose size is greater than 10 MB." In this example, the attribute is the "size" of the sandbox, the condition is "greater than", and the value is "10 MB".

There are two optional delimiter character parameters to the TurboIntegrator function. Because a sandbox has no restrictions on which characters can be used in their name, administrators can supply their own "safe" delimiter when needed.

For example, `ServerSandboxesDelete( 'client:=:Admin, name:=:best case scenario' );"`

In the following example, the colon character is used in the sandbox name ("best::case::scenario") so another delimiter is needed:

```
ServerSandboxesDelete( 'client|=|Admin# name|=|best::case::scenario', '|', '#' );"
```

Note: The exact syntax of a predicate differs between the TurbIntegrator and API forms of this function.

Syntax

```
ServerSandboxesDelete(string,string,string)
```

Arguments

`Predicates`

Required

String

No default

An arbitrary length list of predicates. Each predicate is a string containing three tokens. The first token indicates an attribute of a sandbox. The second indicates a condition, for example ">" or "=". The third token is a possible value of the attribute on which sandboxes should be conditionally filtered. The entire string may not exceed 10,000 characters in length.

`PredicateDelimiter`

Optional

String

default is : (colon)

Optional delimiter character.

The string may not exceed 1 character in length.

`PredicateListDelimiter`

Optional

String

default is , (comma)

Optional delimiter character.

The string may not exceed 1 character in length.

Filter Attributes

Filter attributes are properties of a sandbox on which it can be conditionally matched. Attribute names and their corresponding valid conditions are case insensitive and ignore embedded whitespace. For example, the following two calls are both valid:

```
ServerSandboxesDelete( 'client:=:Admin' );

ServerSandboxesDelete( 'C L I E N T :  =  :Admin' );
```

Table 1. Filter Attributes

Attribute	Description	Valid Conditions	Value Type
UpdateDate	Timestamp of the last write action performed in the sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
AccessDate	Timestamp of the last unload of a sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
CreationDate	Timestamp of the creation of a sandbox.	<, =, >.	Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.

Table 1. Filter Attributes (continued)

Attribute	Description	Valid Conditions	Value Type
Size	The in-memory size of a sandbox.	<, =, >.	Size following log4cxx's conversion rules (see configuration parameter AuditLogMaxTemp FileSize) For example, 10 MB. Kilobytes are the most granular units.
Name	The name of a sandbox.	=, containing.	String.
Client	The owning client of a sandbox.	=.	String.
Group	A group of which the owning client of a sandbox is a member.	=.	String.

Logging and Returns

Sandbox deletion is logged using the preexisting audit logging functionality. Additionally, a more detailed report of the effects of sandbox administration is included in the debug log (tm1server.log) at INFO level. This report will include the list of affected sandboxes, as well as some of their attributes, and any errors encountered.

ServerSandboxesDelete returns only a success or failure status.

Semantics

Predicate List

Multiple predicates passed in a single call to ServerSandboxesDelete are conjunctive. In other words, for a sandbox to match the passed criteria, all predicates must be true. Multiple calls to ServerSandboxesDelete can be used to achieve disjunctive behavior. Only one occurrence of each attribute is allowed per call to ServerSandboxesDelete. For example, passing client twice is invalid as a sandbox has only one owning client. When multiple occurrences of an attribute are detected, a warning displays in the detailed report, however, the operation will not abort in failure. In such a case, the predicates are tested as with any other query, but the results set is always empty.

Locking

To avoid massive locking issues, ServerSandboxesDelete looks at the sandboxes of a client as a point-in-time snapshot and then, when possible, release any locks that would ensure a serializable transaction. Because of this behavior, once a client is "passed" in the iteration of all clients, a sandbox matching the filter criteria may be added to that client before the maintenance transaction completes. This behavior is similar to the behavior that occurs when a sandbox is added to the client immediately after the transaction completes.

Scope

Members of the ADMIN (super-user) and the DataAdmin groups will have access to all sandboxes of all clients. They must explicitly specify the client attribute to limit the scope of their call to ServerSandboxesDelete to only their own sandboxes. All other users have access to only their own sandboxes; if they specify a different client, or a group to which they do not belong, the function will abort in failure and return a privilege error.

In-Use Sandboxes

When a sandbox meets the criteria for deletion, but is currently in use, that sandbox will not be deleted. An entry will appear in the debug log info-level report indicating the occurrence.

Access and Update Dates

Date attributes can be matches with, at most, day granularity. Because of this restriction, recording of these attributes is correspondingly granular. Last Update Date is not updated on individual cell writes. Instead, the system records the unload date of a sandbox that has had something written to it while it was loaded in memory. For such sandboxes, Last Access Date and Last Update Date will be the same. Only Last Access Date is updated on the unloading of a sandbox from memory. Also, because in-memory sandboxes are not subject to ServerSandboxesDelete, Last Access Date is not updated when a sandbox is loaded into memory.

For example, consider the follow usage scenario:

Table 2. Last Access Day Example

Day	Time	Action
1	1	Load Sandbox S
1	2	Write 1
2	3	Read 1
2	4	Unload Sandbox

A user is working with sandbox over the course of two days (perhaps for a much shorter period encompassing the day change.) At time 4, when the sandbox is unloaded, Last Update Date is set to 2, rather than 1 where the last update actually occurred. Last Access Date is also set to 2 at time 4 in this case. If Write1 were instead a read, only Last Access Date would be set to 2, while Last Update Date wouldn't be changed.

Example

```
ServerSandboxesDelete( 'client:=:Admin, name:=:best case scenario' );
```

ServerActiveSandboxGet

This function returns the name of the executing user's active sandbox.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

If the user has no active sandbox, an empty string is returned. Because chores run in the context of a special admin user, and can have no active sandbox, this function will always return an empty string when executed via a chore.

Syntax

```
ServerActiveSandboxGet()
```

Arguments

None.

Example

```
return_value = ServerActiveSandboxGet();
```

This example will return the active sandbox of the user executing the TI process in which the function call is made.

ServerActiveSandboxSet

This function sets the active sandbox of the executing user.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

An empty string is used to clear the executing user's active sandbox. This function will throw an error if the executing user does not own a sandbox with the passed name. Because chores run in the context of a special admin user, and can have no active sandbox, this function will always throw an error when executed via a chore.

Note: For a TurboIntegrator process to read and write values in the context of the executing user's active sandbox, the UseActiveSandbox property must be set. See "GetUseActiveSandboxProperty" on page 257 and "SetUseActiveSandboxProperty."

Syntax

```
ServerActiveSandboxSet(SandboxName)
```

Argument	Description
SandboxName	A string value. The name of a sandbox owned by the executing user.

Example

```
ServerActiveSandboxSet('Best case');
```

This example will set the executing user's active sandbox to "Best case".

```
ServerActiveSandboxSet('');
```

This example will clear the executing user's active sandbox (set context back to the base data.)

SetUseActiveSandboxProperty

This function controls whether a process reads and writes cube data to the base data or to the user's active sandbox.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. The default is for processes to read and write to the base data.

The scope of this function applies only to the current running process and temporarily overrides the permanent value for this property that is set in the Architect / Server Explorer user interface.

Syntax

```
SetUseActiveSandboxProperty(PropertyValue)
```

Argument	Description
PropertyValue	A Boolean value that indicates whether the process should use the active sandbox context when reading and writing cube data. If PropertyValue = 0, the process will disregard the active sandbox context and read/write to the base data. If PropertyValue = 1, the process will read/write cube data to the active sandbox.

Example

```
SetUseActiveSandboxProperty(1);
```

This example will cause the process to read/write cube data to the active sandbox for the rest of this execution.

ServerSandboxExists

This function tests for the existence of the passed sandbox. 1 is returned when the passed sandbox exists, 0 otherwise.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ServerSandboxExists( sandboxname )
```

or

```
ServerSandboxExists( sandboxname , username )
```

Arguments

The name of the sandbox whose existence is being tested. ServerSandboxExists takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

The following snippet shows how the ServerSandboxExists, ServerSandboxGet, and ServerSandboxListCountGet functions can be used to iterate the sandboxes of user called User1 and output those sandboxes to a text file. The TurboIntegrator process would successfully execute for members of the Admin or Data Admin groups and for user called User1. The TurboIntegrator process would fail with a privilege error for any other users.

```
SandboxIndex = 1;
NumSandboxes = ServerSandboxListCountGet( 'User1' );
```

```
WHILE( SandboxIndex <= NumSandboxes );

   SandboxName = ServerSandboxGet( SandboxIndex, 'User1' );

   IF( ServerSandboxExists( SandboxName, 'User1' ) = 1 );

      ASCIIOUTPUT( 'C:\User1Sandboxes.txt', SandboxName );

   ENDIF;

   SandboxIndex = SandboxIndex + 1;

END;
```

ServerSandboxGet

This function returns the name of the sandbox identified by the number *N*, where *N* is the parameter entered.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

`ServerSandboxGet( *index* )`

or

`ServerSandboxGet( *index*, *username*)`

Arguments

The index of the requested sandbox in the user's sandbox collection. The index space will be contiguous, so the first occurrence of an empty string return can be used to break iteration. Also, deleting a sandbox will alter the indexes of any sandboxes that follow that sandbox in the list.

`ServerSandboxGet` takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

See “Example” on page 263

ServerSandboxListCountGet

The function returns the count of sandboxes as a number.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

`ServerSandboxListCountGet()`

or

` ServerSandboxListCountGet( *username* )`

Arguments

`ServerSandboxListCountGet` takes an optional string parameter, the owning client's name. The calling client can use the optional parameter to specify a client other than themselves if the calling client has the appropriate privileges. A privilege error will result if the specified client is not the executing client and the executing client is not a member of the DataAdmin or ADMIN groups. If the optional parameter is not used, the active client's sandboxes are the subject.

Example

See "Example" on page 263

Security TurboIntegrator Functions

These functions pertain to security.

AddClient

This function creates a new client on the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Changes applied through the AddClient functions do not take effect until the Metadata procedure in a process is completed.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
AddClient(ClientName);
```

Argument	Description
ClientName	The name of the client you want to add to the server. The client name is limited to 255 characters/bytes.

Example

```
AddClient('Brian');
```

This example adds the client Brian to the server.

AddGroup

This function creates a new user group on the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Changes applied through the AddGroup function do not take effect until the Metadata procedure in a process is completed.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
AddGroup(GroupName);
```

Argument	Description
GroupName	The name of the group you want to create.

Example

```
AddGroup('Finance');
```

This function adds the Finance user group to the server.

AssignClientToGroup

This function assigns an existing client on a server to an existing user group.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.This function assigns an existing client on a server to an existing user group.

Syntax

```
AssignClientToGroup(ClientName, GroupName);
```

Argument	Description
ClientName	The name of the client you want to assign to a group.
GroupName	The group to which you want to assign the client.

Example

```
AssignClientToGroup('Brian', 'Finance');
```

This example assigns the existing client Brian to the existing user group Finance.

AssignClientPassword

This function assigns a password to an existing client on a server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. AssignClientPassword returns 1 if the password assignment is successful and returns 0 if the assignment fails.

Syntax

```
AssignClientPassword (ClientName, Password);
```

Argument	Description
ClientName	The name of the client for which you want to assign a password.

Argument	Description
Password	The password you want to assign to the client. When assigning a password, use plain text. TM1 will encrypt the password on the server. Passwords must be at least five characters in length.

Example

```
AssignClientPassword ('Brian', 'flyfisher');
```

This example assigns the password 'flyfisher' to the client named Brian.

AssociateCAMIDToGroup

This function creates an association between a TM1 user group and a CAMID.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
AssociateCAMIDToGroup(GroupName, CAMID, CAMIDDisplayValue);
```

Argument	Description
GroupName	The name of the TM1 group you want to associate with the CAMID.
CAMID	The name of the CAMID group. If the CAMID does not exist, it will be created in the }ClientCAMAssociatedGroups control cube.
CAMIDDefDisplayValue	The alias of the CAMID group.

CellSecurityCubeCreate

This function creates a security cube from an existing cube using a reduced set of dimensions.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
CellSecurityCubeCreate ('DataCube', '0:0:1:0');
```

Argument	Description
Cube	Name of the data cube.

Argument	Description
DimensionMap	String specifying whether the dimension at each position should be used in the security cube. The order of dimensions is the original cube order. A 1 for each included dimension and a 0 for an excluded one. Each value separated by a colon.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Creates the cell security cube.

Example

```
CellSecurityCubeCreate ('DataCube', '0:0:1:0');
```

This example creates an RDCLS cube from the cube called Data Cube.

CellSecurityCubeDestroy

This function destroys a security cube that was created from an existing cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
CellSecurityCubeCreate ('DataCube', '0:0:1:0');
```

Argument	Description
Cube	Name of the data cube.
Boolean return	True if the operation succeeded. A major error otherwise.
Additional information	The GrantSecurityAccess property must be set for this TurboIntegrator process to succeed. Destroys the cell security cube.

Example

```
CellSecurityCubeDestroy ('DataCube');
```

DeleteClient

This function deletes a client from the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Changes applied through the DeleteClient function do not take effect until the Metadata procedure in a process is completed.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
DeleteClient(ClientName);
```

Argument	Description
ClientName	The name of the client you want to delete from the server.

Example

```
DeleteClient('Brian');
```

This example removes the client Brian from the server.

DeleteGroup

This function deletes a user group from the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Changes applied through the DeleteGroup function do not take effect until the Metadata procedure in a process is completed.

This function, like all functions that update metadata, should not be used in the Data or Epilog tabs of a process

Syntax

```
DeleteGroup(GroupName);
```

Argument	Description
GroupName	The group you want to delete.

Example

```
DeleteGroup('Finance');
```

This example deletes the Finance user group from the server.

ElementSecurityGet

This function retrieves the security level assigned to a specified group for a dimension element.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ElementSecurityGet(DimName, ElName, Group);
```

Argument	Description
DimName	The parent dimension of the element for which you are retrieving a security level.
ElName	The element for which you are retrieving a security level.
Group	The user group for which you are retrieving a security level.

Example

```
ElementSecurityGet('Region'. 'Germany', 'Budgeting');
```

This example returns the security level assigned to the Budgeting user group for the Germany element of the Region dimension.

ElementSecurityPut

This function assigns a security level to a specified group for a dimension element.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ElementSecurityPut(Level, DimName, ElName, Group);
```

Argument	Description
Level	The security level you are assigning. There are six possible Level values: • None • Read • Write • Reserve • Lock • Admin
DimName	The parent dimension of the element for which you are assigning a security level.
ElName	The element for which you are assigning a security level.
Group	The user group for which you are assigning a security level.

Example

```
ElementSecurityPut('Reserve', 'Region', 'Germany', 'Budgeting');
```

This example assigns Reserve security to the Budgeting group for the Germany element of the Region dimension.

RemoveCAMIDAssociation

This function removes all associations between TM1 user groups and a specified CAMID.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
RemoveCAMIDAssociation(CAMID, RemoveCAMID);
```

Argument	Description
CAMID	The name of the CAMID group for which you want to remove all security associations.
RemoveCAMID	Determines if the specified CAMID is deleted from the }ClientCAMAssociatedGroups control cube. 0 leaves the CAMID in the }ClientCAMAssociatedGroups control cube. 1 deletes the CAMID from the }ClientCAMAssociatedGroups control cube.

RemoveCAMIDAssociationFromGroup

This function removes an association between a TM1 user group and a CAMID.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
RemoveCAMIDAssociationFromGroup(GroupName, CAMID);
```

Argument	Description
GroupName	The name of the TM1 user group for which you want to remove the association.
CAMID	The name of the CAMID group for which you want to remove the association.

RemoveClientFromGroup

This function removes a specified client from a user group.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
RemoveClientFromGroup(ClientName, GroupName);
```

Argument	Description
ClientName	The client you want to remove.
GroupName	The user group from which you want to remove the client.

Example

```
RemoveClientFromGroup('Brian', 'Finance');
```

This example removes the client Brian from the Finance user group.

SecurityRefresh

This function reads all the security control cubes and regenerates the internal structures in the server that are used by TM1 API functions.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SecurityRefresh;
```

Arguments

None.

Server Manipulation TurboIntegrator Functions

These functions facilitate server manipulation.

BatchUpdateFinish

This function instructs the server to exit batch update mode.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

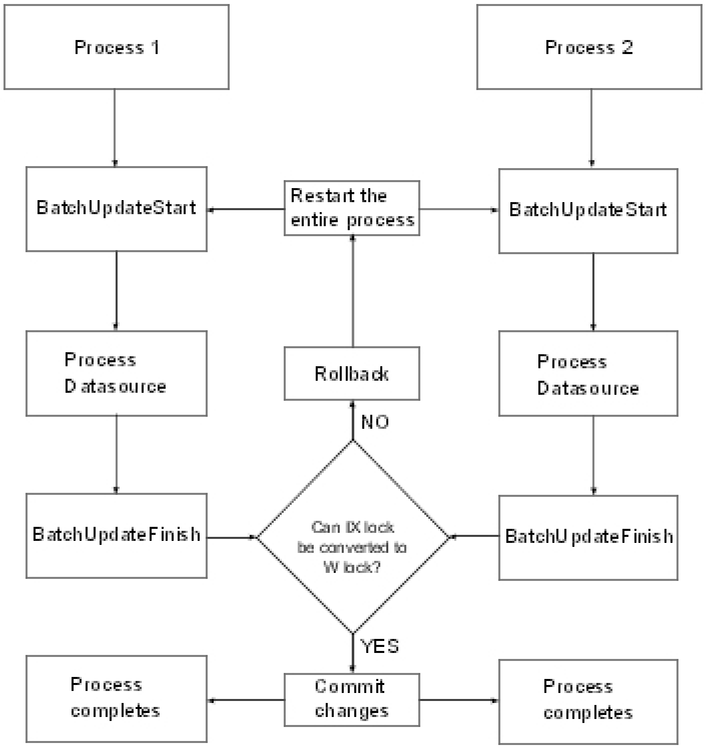
When multiple processes are running in batch update mode and applying changes to a single cube, the TM1 locking scheme may prevent one of the processes from updating the cube. This is by design; when one process obtains a lock to write changes to a cube, other processes will be prevented from writing to that cube in the interest of maintaining data integrity.

This locking scheme can be illustrated using an example of two processes, Process 1 and Process 2, that update a single cube.

- Both processes start and call the BatchUpdateStart function to initiate batch updates.
- Each process operates on a unique data source.
- Process 1 completes processing data and calls the BatchUpdateFinish function. The process obtains a write lock to the cube and commits changes.
- While Process 1 still holds a write lock to the cube, Process 2 completes processing data and calls the BatchUpdateFinish function. However, because

Process 1 retains the lock, Process 2 cannot obtain a lock to the cube. All data changes applied in Process 2 are rolled back and Process 2 is restarted. This ensures data integrity.

Syntax



Depending on the size of the datasource for Process 2, the data rollback and process re-execution can cause a noticeable decrease in performance. To address this performance issue, consider using the BatchUpdateFinishWait function in place of BatchUpdateFinish.

```
BatchUpdateFinish(SaveChanges);
```

Argument	Description
SaveChanges	A flag that instructs the server to either save or discard changes committed while in batch update mode. Specify 0 to save changes, 1 to discard changes.

Example

```
BatchUpdateFinish(0);
```

This example instructs the server to save changes to TM1 data and exit batch update mode.

BatchUpdateFinishWait

If a process calls BatchUpdateFinishWait, but is unable to secure a cube write lock to commit changes, the process will wait until the lock becomes available and then commit changes.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This TurboIntegrator function is identical to the BatchUpdateFinish function with the following exception: the process will wait until the lock becomes available and then commit changes.

Data changes applied in the process are not rolled back and the process is not re-executed.

Note: While waiting for the cube write lock, the process releases any read locks it acquired for other objects during process execution. Because these read locks are released before the process can commit changes to the cube, the objects for which the read locks are released can be modified *before* the cube is updated. This can lead to data inconsistency when using BatchUpdateFinishWait.

We recommend that BatchUpdateFinishWait be used only in controlled situations where you know that other processes are not modifying data or metadata related to the process that calls BatchUpdateFinishWait.

Syntax

```
BatchUpdateFinishWait(SaveChanges);
```

Argument	Description
SaveChanges	A flag that instructs the server to either save or discard changes committed while in batch update mode. Specify 0 to save changes, 1 to discard changes.

Example

```
BatchUpdateFinishWait(0);
```

This example instructs the server to save changes to TM1 data and exit batch update mode.

BatchUpdateStart

This function enables batch updates.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
BatchUpdateStart;
```

Arguments

None.

CubeSaveData

CubeSaveData() serializes a cube.

Syntax

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

To improve performance, transaction logging may be disabled while loading data. To safeguard newly loaded data in the unlikely event of a server crash, the changes can be serialized to disk. SaveDataAll has been used to serialize data to disk and to truncate the transaction log. When processing a SaveDataAll command, the server acquires a READ lock on every cube and an IX lock on every changed cube. This can cause significant contention with user activity if SaveDataAll is run during periods of user activity.

Typically not all the cubes affected by SaveDataAll need to be serialized since not all cubes are typically loaded with new data. CubeSaveData is used to serialize an individual cube to disk. CubeSaveData serializes the cube's data that has been committed to memory including the modifications that have been performed against it in the current TurboIntegrator process but not yet committed.

```
CubeSaveData(Cube);
```

Argument	Description
Cube	The name of the cube you want to serialize.

Example

```
CubeSaveData ('SalesCube');
```

Consider the following TurboIntegrator process code:

```
CellPutN(500, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');

CubeSaveData('y2ksales');

CellPutN(1000, 'y2ksales', 'Actual', 'Argentina', 'S Series 1.8 L Wagon', 'Sales', 'Jan');
```

When the CubeSaveData command is processed, the value of 500 for the January Sales cell will be included in the cube's serialization to disk, even though it has not yet been committed. The update of the January Sales cell to 1000 will not be part of the serialization.

Transaction Log

A new transaction entry appears in the Transaction log when CubeSaveData has been run. When processing a transaction log file during recovery, all updates to a cube that have been applied so far will be discarded when a CubeSaveData directive against the cube is encountered as all of the updates have already been serialized to the cube.

Server Crash Recovery

The SaveDataAll command takes advantage of the fact that all cubes are locked during its processing and truncates the transaction log knowing that all updates performed before serialization have been safely stored to disk. This is not the case for CubeSaveData so you must modify the way data recovery is performed when a cube has been serialized.

The transaction log file could contain records that represent changes that are older than the most recent data in the cube and should not be applied when data is being recovered.

DisableBulkLoadMode

Used to disable bulk load processing.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes. See “EnableBulkLoadMode” for details.

EnableBulkLoadMode

Enables Bulk Load Mode for a TurboIntegrator process.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

You can enable Bulk Load Mode in either the Prolog or Epilog section of a TurboIntegrator process. For efficiency, enable Bulk Load Mode in the first, or very close to the first, statement in the Prolog section of your process.

After enabling Bulk Load Mode in a process, it can only be disabled on the last line in the Epilog section. If you attempt to disable Bulk Load Mode anywhere else in the process, the process will not compile.

If the mode is enabled in one TurboIntegrator process, it remains enabled until explicitly disabled or until the chore completes. This means you can enable the mode in a process within a chore and then run a series of TurboIntegrator processes before disabling it. You can also enter and exit Bulk Load Mode repeatedly, using the mode only for certain critical parts of a chore.

Use the following TurboIntegrator commands to enable and disable Bulk Load Mode in a TurboIntegrator process.

```
EnableBulkLoadMode()
```

`DisableBulkLoadMode()` - This function can only be used on the last line in the Epilog section of your TI process when using Bulk Load Mode.

SaveDataAll

This function saves all TM1 data from server memory to disk and restarts the log file.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Using SaveDataAll in a Chore

SaveDataAll commits all changes a chore makes prior to calling the SaveDataAll function.

While a chore is running, it accumulates locks on the objects it accesses. The commit operation initiated by the SaveDataAll function temporarily releases all these locks. Once the commit is complete, SaveDataAll reacquires all the locks it had before so it can continue to access the objects it was working on.

There is a brief window during the commit operation where the locks are released and another user or TurboIntegrator process could delete objects the original chore was using. When the original chore attempts to reacquire the locks on those objects, the objects will not be available and the chore will cease processing. In this case, an error similar to the following is written to the Tm1s.log file:

```
844 WARN 2008-04-01 16:40:09,734 TM1.Server TM1ServerImpl::FileSave
could
not reacquire lock on object with index 0x200002ca
```

Lock contention and using SaveDataAll at the end of TurboIntegrator processes

Using SaveDataAll as last command in a TurboIntegrator process can increase lock contention in TM1 TurboIntegrator processes.

In IBM Cognos TM1 versions, SaveDataAll was often added to the end of a TurboIntegrator process that loads data with logging disabled. The SaveDataAll provided a way to write data from memory to disk directly after a successful import, so that the newly imported data would not be lost in case of a mishap, such as a server crash.

However, adding SaveDataAll as the last command can result in numerous TurboIntegrator import processes, each one with SaveDataAll as last command. This technique worked in TM1 Version 9.0 and older due to the previous lock model which used only the global write lock. At any given time in earlier versions only one write operation could take place. Therefore competing concurrent SaveDataAll operations never occurred from multiple concurrent write operations.

Version 9.1 and newer introduced a more granular lock-by-object model that enables concurrent write operations, if these write operations do not compete for the same resources. If they do compete for the same resources, a lock contention occurs forcing one of the processes to rollback. So now two TurboIntegrator import processes may run simultaneously if they do not share any objects, for example, if they import into two different cubes.

The TurboIntegrator function SaveDataAll relies on the transaction logfile tm1s.log and involves all objects within a data model. Therefore, two TurboIntegrator import processes, both using the function SaveDataAll, cannot run in parallel: one will be executed, the other one (and its TurboIntegrator process) will be forced to rollback. The same is true if the TurboIntegrator processes are part of chores: only one chore will proceed to execute the TurboIntegrator function SaveDataAll, the other chore will be forced to rollback.

A rollback is undesirable from a performance point of view, as it increases the total execution time of a TurboIntegrator process or chore. Competing concurrent SaveDataAll operations will always lead to a lock contention and to a rollback.

There are two possible solutions to avoid competing concurrent SaveDataAll operations:

- Do not use the TurboIntegrator function SaveDataAll. Instead enable Cube Logging for the import cubes.

- If enabling Cube Logging for the import cubes cannot be done for performance reasons, within the TM1 application there should be only one process calling the TurboIntegrator function SaveDataAll. Use a stand-alone, single, distinct chore to execute the SaveDataAll operation.

Syntax

```
SaveDataAll;
```

Arguments

None.

ServerShutdown

This function shuts down a server running as an application. ServerShutdown cannot be used to shut down a server running as a Windows service.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ServerShutDown(SaveData);
```

Argument	Description
SaveData	A Boolean value that indicates whether the server should save changes to disk before shutting down. If SaveData = 0, the server shuts down without saving changes. If SaveData = 1, the server saves changes from memory to disk before shutting down.

Example

```
ServerShutdown(1);
```

This example shuts down the server and saves data to disk.

Subset Manipulation TurboIntegrator Functions

These functions facilitate subset manipulation.

SubsetAliasSet

This function sets the alias attribute to be used in a subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

SubsetAliasSet returns 1 if successful, 0 otherwise.

Syntax

```
SubsetAliasSet( DimName, SubName, AliasName );
```

Argument	Description
DimName	The parent dimension of the subset for which you want to set the alias.
SubName	The subset for which you want to set the alias.
Aliasname	The alias you want to use in the subset.

SubsetCreate

This function creates an empty public subset of a specified dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetCreate(DimName, SubName);
```

Argument	Description
DimName	The parent dimension of the subset you are creating.
SubName	The name you want to assign to the subset.

Example

```
SubsetCreate('Region', 'Northern Europe');
```

This example creates the empty Northern Europe subset of the Region dimension. You can use SubsetElementInsert to add elements to the subset.

SubsetCreateByMDX

This function creates a public subset based on a passed MDX expression.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetCreatebyMDX(SubName, MDX_Expression);
```

Argument	Description
SubName	The name you want to assign to the subset.
MDX_Expression	An MDX expression that returns a subset.

Example

```
SubsetCreatebyMDX('0-level months', '{TM1SORT( {TM1FILTERBYLEVEL( {TM1SUBSETALL([month] )}, 0)}, A
```

This example creates a public subset named '0-level months' based on an MDX expression that returns a subset consisting of all 0-level elements in the Month dimension, sorted in ascending alphabetical order.

SubsetDeleteAllElements

This function deletes all elements from a public subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetDeleteAllElements(DimName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete elements.
SubsetName	The subset from which you want to delete elements. This must be a public subset. TurboIntegrator cannot access private objects.

Example

```
SubsetDeleteAllElements('Region', 'Central Europe');
```

This example deletes all elements from the Central Europe subset of the Region dimension.

SubsetDestroy

This function deletes a subset from the TM1 database.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetDestroy(DimName, SubName);
```

Argument	Description
Dimname	The parent dimension of the subset you are deleting.
SubName	The name of the subset you want to delete.

Example

```
SubsetDestroy('Region', 'Northern Europe');
```

This example deletes the Northern Europe subset of the Region dimension.

SubsetElementDelete

This function deletes an element to a subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetElementDelete(DimName, SubName, Index);
```

Argument	Description
DimName	The parent dimension of the subset from which you want to delete an element.
SubName	The subset from which you want to delete an element.
Index	The index number of the element you want to delete from the subset.

Example

```
SubsetElementDelete('Region', 'Northern Europe', 3);
```

This example deletes the third element from the Northern Europe subset of the Region dimension.

SubsetElementInsert

This function adds an element to an existing subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetElementInsert(DimName, SubName, ElName, Position);
```

Argument	Description
DimName	The parent dimension of the subset to which you want to add an element.
SubName	The name of the subset to which you are adding an element.
ElName	The name of the element you want to add to the subset. The element must exist in the TM1 database.
Position	A value that indicates the index position of the element within the subset.

Example

```
SubsetElementInsert('Region', 'Northern Europe', 'Finland',3);
```

This example adds the element Finland to the Northern Europe subset of the Region dimension. Finland is the third element in the subset definition.

SubsetExists

Use SubsetExists to determine if a specific public subset exists on the server from which a TurboIntegrator process is executed.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The function returns 1 if the subset exists on the server, otherwise it returns 0. Note that this function cannot be used to determine the existence of private subsets.

Syntax

```
SubsetExists(DimName, SubsetName);
```

Argument	Description
DimName	The name of the dimension that is the parent of the subset whose existence you want to confirm.
SubsetName	The name of the public subset whose existence you want to confirm

Example

```
SubsetExists('Region', 'Northern Europe');
```

This example determines if Northern Europe subset of the Region cube exists on the server.

SubsetExpandAboveSet

This function sets the Expand Above property for a subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

When this property is set to TRUE, children of a consolidation are displayed above the consolidation when the consolidation displays on a row, and to the left of the consolidation when the consolidation displays on a column.

The function returns 1 if successful, otherwise it returns 0.

Syntax

```
SubsetExpandAboveSet( DimName, SubsetName, ExpandAboveFlag
);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to set the Expand Above property.

Argument	Description
SubsetName	The subset for which you want to set the Expand Above property.
ExpandAboveFlag	Set ExpandAboveFlag to 1 to set the Expand Above property to TRUE. When this property is TRUE, consolidations expand above on rows and to the left on columns. Set ExpandAboveFlag to 0 to set the Expand Above property to FALSE. When this property is FALSE, consolidations expand below on rows and to the right on columns.

Example

```
SubsetExpandAboveSet('Region', 'Europe', 1 );
```

This example sets the Expand Above property to TRUE for the Europe subset of the Region dimension.

SubsetFormatStyleSet

This function applies an existing display style to a named subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Display styles are defined for specific elements. If you apply an existing display style to a subset that includes elements that are not included in the display style, no formatting is applied to those elements.

Syntax

```
SubsetFormatStyleSet( DimName, SubsetName, FormatName
);
```

Argument	Description
DimName	The parent dimension of the subset to which you want to apply a display style.
SubsetName	The name of the subset to which you are applying a display style.
FormatName	The name of the existing display style you want to apply to the subset.

Example

```
SubsetFormatStyleSet ('Region', 'Northern Europe', 'BoldCurrencyLeftJustified');
```

This example applies the BoldCurrencyLeftJustified display style to the Northern Europe subset of the Region dimension.

SubsetGetElementName

This function returns the name of the element at a specified index location within a given subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetGetElementName(DimName, SubsetName, ElementIndex);
```

Argument	Description
DimName	The parent of the subset from which you want to retrieve an element name.
SubsetName	The subset from which you want to retrieve an element name.
ElementIndex	A number representing the position within the subset of the element you want to retrieve.

Example

```
SubsetGetElementName('Region', 'Americas', 4);
```

This example returns the name of the fourth element in the Americas subset of the Region dimension.

SubsetGetSize

This function returns the number of elements in a subset.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
SubsetGetSize(DimName, SubsetName);
```

Argument	Description
DimName	The parent dimension of the subset for which you want to determine size.
SubsetName	The subset for which you want to determine size.

Example

```
SubsetGetSize('Region', 'EurAsia');
```

This function returns the number of elements in the EurAsia subset of the Region dimension.

SubsetIsAllSet

This function sets a subset to use all elements of the parent dimension.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

It is equivalent to clicking the on the Subset Editor. SubsetIsAllSet returns 1 if successful, 0 otherwise.

Syntax

```
SubsetIsAllSet( DimName, SubName, Flag );
```

Argument	Description
DimName	The parent dimension of the subset for which you want to use all elements.
SubName	The subset for which you want to use all dimension elements.
Flag	Any non-zero value specifies that the subset uses all the current elements from the parent dimension and will dynamically update to use all elements from the parent dimension whenever the subset is called. Specifying a zero value freezes the elements in the subset as the current set of all elements in the parent dimension. The subset will not dynamically update to use all dimension elements in the future.

View Manipulation TurboIntegrator Functions

These functions pertain to view manipulation.

PublishView

This function publishes a named private view on the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
PublishView(Cube, View, PublishPrivateSubsets, OverwriteExistingView);
```

Argument	Description
Cube	The name of the cube containing the private view to be published.
View	The name of the private view to be published.

Argument	Description
PublishPrivateSubsets	This Boolean argument (1 or 0) determines if any private subsets present in the view should also be published. If PublishPrivateSubsets is true (1) , all private subsets used in the view are published along with the view. If this argument is false (0) , private subsets are not published. A public view cannot contain private subsets, so the view will not be published and an error will be written to the TurboIntegrator log file. **Note:** If a private subset contains another private subset as a user-defined consolidation, the subset can never be published using the PublishView function, regardless of the value of the PublishPrivateSubsets argument.
OverwriteExistingView	This Boolean argument (1 or 0) determines if any existing identically named public view should be overwritten when the private view is published. If OverwriteExistingView is true (1) , any existing identically named public view will be overwritten when the private view is published. If this argument is false (0), the public view will not be overwritten, the private view will not be published, and an error will be written to the TurboIntegrator log file.

ViewColumnDimensionSet

This function sets a column dimension for a TM1 view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewColumnDimensionSet(CubeName, ViewName, DimName, StackPosition);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the column dimension.
ViewName	The view for which you are setting the column dimension.
DimName	The dimension you want to set as a column dimension for the view.

Argument	Description
StackPosition	A number that indicates the stack position of the dimension in the view. This is a 1-based number. 1 indicates the top-most stack position. 2 indicates a position below 1, and so on.

Example

```
ViewColumnDimensionSet('98sales', 'Quarter1', 'Month',1);
```

This example sets Month as a column dimension for the 1Quarter view of the 98sales cube. In the event of stacked column dimensions, Month is placed in the top-most position.

ViewColumnSuppressZeroesSet

This function suppresses or enables the display of columns containing only zero values in a TM1 cube view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewColumnSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of columns containing only zeroes in the view. Specify 0 to enable the display of columns containing only zeroes.

Example

```
ViewColumnSuppressZeroesSet('99sales', '1st QuarterActuals', 1);
```

This example suppresses the display of any columns containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewConstruct

This function constructs, pre-calculates, and stores a stargate view in memory on a server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

This function is useful for pre-calculating and storing large views so they can be quickly accessed after a data load or update.

Syntax

```
ViewConstruct(CubeName, ViewName);
```

Argument	Description
CubeName	The cube from which you want to construct the view.
ViewName	The view you want to construct. This view must be an existing public view on the server.

Example

```
Viewconstruct('99sales', '1st Quarter Actuals')
```

This example deletes the third element from the Northern Europe subset of the Region dimension.

ViewCreate

This function creates an empty view of a specified cube.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Note: If you want to perform a replication or synchronization operation after using the ViewCreate function in a TI process, call the `SaveDataAll`function from the Epilog procedure of the process to make sure the newly created view is available for the replication. These steps apply only when you use the ViewCreate function before a replication or synchronization operation.

Syntax

```
ViewCreate(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you are creating.
ViewName	The name you want to assign to the view.

Example

```
ViewCreate('99sales', '1st Quarter Actuals');
```

ViewDestroy

This function deletes a view from the TM1 database.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewDestroy(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you are deleting.
ViewName	The name of the view you want to delete.

Example

```
ViewDestroy('99sales', '1st Quarter Actuals');
```

This example deletes the 1st Quarter Actuals view of the 99sales cube.

ViewExists

Use ViewExists to determine if a specific public view exists on the server from which a TurboIntegrator process is executed.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The function returns 1 if the view exists on the server, otherwise it returns 0. Note that this function cannot be used to determine the existence of private views.

Syntax

```
ViewExists(CubeName, ViewName);
```

Argument	Description
CubeName	The name of the cube that is the parent of the view whose existence you want to confirm.
ViewName	The name of the public view whose existence you want to confirm

Example

```
ViewExists('Inventory', 'FebClosing');
```

This example determines if FebClosing view of the Inventory cube exists on the server.

ViewExtractSkipCalcsSet

This function sets an option to include/exclude consolidated values in a view **and** any associated view extracts.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

ViewExtractSkipCalcsSet is the equivalent of the Skip Consolidated Values option in the View Extract dialog box.

Syntax

```
ViewExtractSkipCalcsSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.
ViewName	The view for which you are setting the option.
Flag	A binary value that turns the option on or off. Specify 1 to exclude consolidated values from the view extract. Specify 0 to include consolidated values.

Example

```
ViewExtractSkipCalcsSet ('99sales', '1st Quarter Actuals',1);
```

This example turns on the Skip Consolidated Values option for the 1st Quarter Actuals view. The view extract will not include any consolidated values.

ViewExtractSkipRuleValuesSet

This function sets an option to include/exclude rule-calculated values in a view **and** any associated view extracts.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

ViewExtractSkipRuleValuesSet is the equivalent of the Skip Rule Calculated Values option in the View Extract dialog box.

Syntax

```
ViewExtractSkipRuleValuesSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the option.
ViewName	The view for which you are setting the option.

Argument	Description
Flag	A binary value that turns the option on or off. Specify 1 to exclude rule-calculated values from the extract. Specify 0 to include rule-calculated values.

Example

```
ViewExtractSkipRuleValuesSet ('99sales', '1st QuarterActuals', 1);
```

This example turns on the Skip Rule Calculated Values option for the extract created from the 1st Quarter Actuals view. The extract will not include any rule-calculated values.

ViewExtractSkipZeroesSet

This function sets an option to include/exclude zero values in a view **and** any associated view extracts.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

A view extract is a TM1 view exported as an ASCII comma-delimited (.cma) file.

Note: This function affects views as they exist on the server. The scope of this function is not restricted to extracts generated from a view.

ViewExtractSkipZeroesSet is the equivalent of the Skip Zero/Blank Values option in the View Extract dialog box.

Syntax

```
ViewExtractSkipZeroesSet (Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you are setting the Skip Zeroes option.
ViewName	The view for which you are setting the Skip Zeroes option.
Flag	A binary value that turns the option on or off. Specify 1 to exclude zeroes from the extract. Specify 0 to include zeros.

Example

```
ViewExtractSkipZeroesSet ('99sales', '1st Quarter Actuals',1);
```

This example turns on the Skip Zeroes option for the extract created from the 1st Quarter Actuals view. The extract will not include any zero or blank values.

ViewRowDimensionSet

This function sets a row dimension for a TM1 view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewRowDimensionSet(CubeName, ViewName, DimName, StackPosition);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the row dimension.
ViewName	The view for which you are setting the row dimension.
DimName	The dimension you want to set as a row dimension for the view.
StackPosition	A number that indicates the stack position of the dimension in the view. This is a 1-based number. 1 indicates the left-most stack position. 2 indicates a position to the right of 1, and so on. **Note:** It is possible for a TM1 client to set a Tm1p.ini parameter (BrowseDisplayReadsRightToLeft=T) that reverses the orientation of data in the Cube Viewer. When the orientation of data is reversed, the stack positions are also reversed. 1 indicates the right-most stack position. 2 indicates a position to the left of 1, and so on.

Example

```
ViewRowDimensionSet('98sales', 'Quarter1', 'Month',1)
```

This example sets Month as a row dimension for the 1Quarter view of the 98sales cube. In the event of stacked row dimensions, Month is placed in the left-most position.

ViewRowSuppressZeroesSet

This function suppresses or enables the display of rows containing only zero values in a TM1 cube view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewRowSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of rows containing only zeroes in the view. Specify 0 to enable the display of rows containing only zeroes.

Example

```
ViewRowSuppressZeroesSet('99sales', '1st Quarter Actuals',1);
```

This example suppresses the display of any rows containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewSubsetAssign

This function assigns a named subset to a cube view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewSubsetAssign(Cube, ViewName, DimName, SubName);
```

Argument	Description
Cube	The parent cube of the view to which you are assigning a subset.
ViewName	The view to which you are assigning a subset.
DimName	The parent dimension of the subset you are assigning to the view.
SubName	The name of the subset you want to assign to the view.

Example

```
ViewSubsetAssign('99sales', '1st Quarter Actuals', 'Month','Q1');
```

This example assigns the Q1 subset of the Month dimension to the 1st Quarter view.

ViewSuppressZeroesSet

This function suppresses or enables the display of all rows and columns containing only zero values in a TM1 cube view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewSuppressZeroesSet(Cube, ViewName, Flag);
```

Argument	Description
Cube	The parent cube of the view for which you want to suppress or enable the display of zero values.
ViewName	The view for which you want to enable or suppress the display of zeroes.
Flag	A binary value that enables or suppresses zeroes. Specify 1 to suppress the display of rows or columns containing only zeroes in the view. Specify 0 to enable the display of rows and columns containing only zeroes.

Example

```
ViewSuppressZeroesSet('99sales', '1st Quarter Actuals',1);
```

This example suppresses the display of any rows or columns containing only zeroes in the 1st Quarter Actuals view of the 99sales cube.

ViewTitleDimensionSet

This function sets a title dimension for a TM1 view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewTitleDimensionSet(CubeName, ViewName, DimName);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the title dimension.
ViewName	The view for which you are setting the title dimension.
DimName	The dimension you want to set as a title dimension for the view.

Example

```
ViewTitleDimensionSet('98sales', 'Quarter1', 'Month');
```

This example sets Month as a title dimension for the 1Quarter view of the 98sales cube.

ViewTitleElementSet

This function sets a title element for a TM1 view.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

ViewTitleElementSet is used in conjunction with the ViewTitleDimensionSet function.

Syntax

```
ViewTitleElementSet(CubeName, ViewName, DimName, Index);
```

Argument	Description
CubeName	The parent cube of the view for which you are setting the title element.
ViewName	The view for which you are setting the title element.
DimName	The parent dimension of the title element.
Index	An index into the specified dimension that indicates the element to be set as the title element.

Example

```
ViewTitleElementSet('98sales', 'Quarter1', 'Model',3);
```

This example sets the third element of the Model dimension as a title element for the Quarter1 view of the 98sales cube.

ViewZeroOut

This function sets all data points in a view to zero.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
ViewZeroOut(Cube, ViewName);
```

Argument	Description
Cube	The parent cube of the view you want to zero out.
ViewName	The view you want to zero out.

Example

```
ViewZeroOut('99sales', '1st Quarter Actuals');
```

This example sets all data points in the 1st Quarter Actuals view to zero.

Miscellaneous TurboIntegrator Functions

These functions facilitate miscellaneous tasks.

AddInfoCubeRestriction

This TurboIntegrator function filters InfoCube data as it is pulled into TM1 .

Use this function to restrict the values that are imported for a specified characteristic. This function must be placed in the Prolog. The function can be called multiple times to filter more than one characteristic in a single process.

Syntax

```
AddInfoCubeRestriction(STRING CharactName, STRING sign,STRING compOperator,
STRING lowValue, STRING highValue)
```

Argument	Description
STRING CharactName	Contains the technical name of the characteristic to be restricted. The data type has to be a character string with a length equal to or less than 30.
STRING sign	Contains either I (= inclusive) or E (= exclusive). Exclusive is the logical NOT for the restriction specified by this row. The data type has to be a character of length 1.
STRING compOperator	Contains the relational comparative operator. The data type has to be a character string of length 2. Valid comparative operators are: 'EQ' = equal 'NE' = not equal 'LT' = less than 'GT' = grater than 'LE' = less or equal 'GE' = grater or equal 'BT' = between 'NB' = not between
STRING lowValue	Contains the low value for the operator specified in the row before. The data type has to be a character string with a length equal to or less than 60.

Argument	Description
STRING highValue	Contains the high value for the operator specified two rows before. The data type has to be a character string with a length equal to or less than 60. It is only needed for the operators BT and NB, otherwise it is ignored, and in this case an empty string should be placed here.

Example

```
AddInfoCubeRestriction('0CALYEAR','E','BT','1997','2000');
```

returns all characteristic values between 1997 and 2000.

```
AddInfoCubeRestriction('0CALYEAR','I','NB','1997', '2000') ;
```

returns all characteristic values not between 1997 and 2000.

```
AddInfoCubeRestriction('0DOC_CURRCY', 'I', 'NE', 'USD','') ;
```

returns all characteristic values not equal to USD.

DataSourceSAPUsingRoleAuths

This TurboIntegrator function instructs the TurboIntegrator process to ignore security information when processing an SAP datasource. This variable must be placed in the Prolog.

Syntax

```
DataSourceSAPUsingRoleAuths='0'
```

Argument	Description
0	Security information is *ignored* when processing an SAP datasource.
1	Security information is *read* when processing an SAP datasource.

DataSourceSAPUsingTexts

This TurboIntegrator local variable instructs the TurboIntegrator process to ignorecharacteristic descriptions when processing an SAP datasource, resulting in a decreased memory consumption and increased performance. This variable must be placed in the Prolog.

Syntax

```
DataSourceSAPUsingTexts='0'
```

Argument	Description
0	Characteristic descriptions *are ignored* when processing an SAP datasource. The characteristic technical name is imported into TM1 as both an element name and alias.

Argument	Description
1	Characteristic descriptions *are read* when processing an SAP datasource.

Expand

This function "expands" TurboIntegrator variable names, enclosed in % signs, to their values at run time.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

If the variable name represents a string variable, the entire variable expression must be enclosed on quotes. For example, "%V1%".

A common use of the Expand function is to pass the value of TurboIntegrator variables to the ODBCOutput function. Refer to the example below for details.

If Expand is fed with a numerical value, an implicit type conversion is performed and the numerical value is converted into a string.

That string has a fixed minimum length of 10 characters. If the converted number is too small to fill 10 characters, it is padded with leading spaces. Only three leading decimal characters are converted. For example, a numerical value of 0.123456789 is converted into the string "0.123".

Syntax

```
Expand(String);
```

Argument	Description
String	Any string that includes TurboIntegrator variable names enclosed in % signs.

Example

```
ODBCOutPut( 'TransData', Expand( 'INSERT INTO SALES( MONTH, PRODUCT, SALES )
VALUES ( "%V0%", "%V1%",%V2% )' ) );
```

This example illustrates the use of the Expand function within the ODBCOutput function. The example inserts records into a relational table named Sales that consists of three columns: Month, Product, and Sales.

The Expand function converts the variables V0, V1, and V2 to their actual values within the view. Assuming that the first value in the view is 123.456, and is defined by the elements Jan and Widget

```
Expand( 'INSERT INTO SALES ( MONTH, PRODUCT, SALES ) VALUES ("%V0%", "%V1%",%V2% )' )
```

becomes

```
'INSERTINTO SALES ( MONTH, PRODUCT, SALES ) VALUES ( Jan, Widget,123.456 )'
```

at run time.

FileExists

This function determines if a specified file exists. The function returns 1 if the file exists, 0 if it does not.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
FileExists(File);
```

Argument	Description
File	The name of a file. If a full parth is not specified, TM1 searches for the file in the server data directory.

Example

```
FileExists('C:\tm1s7\pdata\model.dim');
```

This example determines if the model.dim file exists.

NumberToString

This function converts a number to a string, using the decimal separator for the current user locale.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

In Microsoft Windows, the decimal separator is a Regional Options setting.

The output of this function is similar to the 'general' number format; it does not use thousands separators and uses the minus sign (-) to denote negative numbers.

Syntax

```
NumberToString(Value);
```

Argument	Description
Value	The real value that you want to convert to a string.

Example

```
nRET = NumberToString(1234.5);
```

NumberToStringEx

This function converts a number to a string, using the passed string format, decimal separator, and thousands separator.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

```
NumberToStringEx(Value, NumericFormat, DecimalSep, ThousandsSep);
```

Argument	Description
Value	The real value that you want to convert to a string.
FormatString	A TM1 numeric format string that defines the format for the function output. Numeric formats are described in the IBM Cognos TM1 *User Guide*.
DecimalSep	The decimal separator to be used in the output string.
ThousandsSep	The thousands separator to be used in the output string.

Example

```
sRet=NUMBERTOSTRINGEX(7895.23,'#,0.#########', ',','.');
ASCIIOUTPUT('number_to_string.txt',sRet);
```

Will return in ascii file;

7.895,23

RefreshMdxHierarchy

This function updates the MDX hierarchies in a server without requiring you to restart the server.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

Syntax

Use this function after configuring or editing the custom named hierarchy levels for a dimension in the }HierarchyProperties control cube.

For details on using named levels with dimensions, see the related section in the IBM Cognos TM1 *Developer Guide*.

```
RefreshMdxHierarchy(dimensionName)
```

Argument	Description
dimensionName	Optional string parameter to specify a specific dimension to update. Leave this parameter blank to update all dimensions.

Example

Update all dimensions:

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

To update only the customers dimension:

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

StringToNumber

This function converts a string to a number, using the decimal separator for the current user locale.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

In Microsoft Windows, the decimal separator is a Regional Options setting.

If the input string is an invalid number string, the value returned will be an invalid floating point value.

Syntax

```
StringToNumber(String);
```

Argument	Description
String	The string you want to convert to a number.

Example

```
nRET = StringToNumber('123.45');
```

StringToNumberEx

This function converts a string to a number, using the passed decimal separator and thousands separator.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

If the input string is an invalid number string, the value returned will be an invalid floating point value.

Syntax

```
StringToNumberEx(String, DecimalSep, ThousandsSep);
```

Argument	Description
String	The string that you want to convert to a number.
DecimalSep	The decimal separator to be used in the output number.
ThousandsSep	The thousands separator to be used in the output number.

Example

```
nRET = StringToNumberEx('12453.45', ' . ', ' , ');
```

TM1ProcessError.log file

When a TurboIntegrator process encounters an error, it generates a TM1ProcessError.log file. This log file is saved to the data directory of the server on which the process resides.

A TM1ProcessError.log file contains a list of errors encountered by the process. For each error encountered, the log file records the tab and line that caused the error, along with a brief description of the error.

When a process error log file is generated, TM1 assigns a unique name that lets you readily identify which TurboIntegrator process generated the error file and the time at which the file was created. File names are assigned using the convention

TM1ProcessError_<*time stamp*>_<*process name*>.log. In this convention, <*time stamp*> is the time (expressed as yyyymmddhhmmss GMT) at which the file was generated and <*process name*> is the name of the TurboIntegrator process that caused the errors.

For example, an error file named TM1ProcessError_20040224203148_ CreateSalesCube.log indicates that the error file was generated at 20:31:48 GMT on February 24, 2004 and that it contains errors caused by the CreateSalesCube process.

TM1User()

This function returns a string giving the current TM1 Client.

When executed in a process that the user is running directly, it will return the user's TM1 client name. When executed in a chore that the user runs directly, it will also return the user's TM1 client name.

If run from a scheduled chore, it will return a name in the form *R*<chore name>*, for example, `R*UpdateRegionDimension`.

Syntax

```
TM1User()
```

WildcardFileSearch

This function lets you use wildcard characters to search for files in a specified directory.

This is a TM1 TurboIntegrator function, valid only in TurboIntegrator processes.

The results of the WildCardFileSearch function may vary depending on the operating system in use. Files in a Windows directory are sorted in alphabetical order while files in a UNIX directory are sorted in random order. Because the order of sorting varies between the operating systems, the identical WildCardFileSearch function executed against identical directories, one on Windows and one on UNIX, will yield different results.

Syntax

```
WildcardFileSearch( Pathname, PriorFilename);
```

Argument	Description
Pathname	A pathname to files for which you want to search. The pathname must end in a filename, which can contain a wildcard sequence using the * and/or ? characters. The ? wildcard character matches any single character. The * wildcard character matches zero or more characters.
PriorFilename	The name of an existing file in the specified directory. This filename cannot contain wildcard characters. The wildcard search specified by the Pathname argument will commence *AFTER* this file. If you pass an empty string as the PriorFilename argument, the WildcardFileSearch function returns the first file that matches the wildcard sequence specified by the Pathname argument.

Example

The following example shows the use of the WildcardFileSearch function to determine the first server log file generated in 2004:

```
file = WildcardFileSearch( 'C:\Program Files\Cognos\TM1\Custom\
TM1Data\SData\tm1s2004*.log', ' ');
```

This example returns the first file matching the wildcard sequence 'tm1s2004*.log' from the C:\Program Files\Cognos\TM1\Custom\TM1Data\SData\ directory.

Because server log files are named and saved with sequential time stamps, and because the second parameter to WildcardFileSearch is empty, the function returns the first server log file starting with the characters 'tm1s2004'. This would be the first server log file generated in the year 2004.

The following example shows the use of the WildcardFileSearch function to return the first server log file generated after tm1s20040211153827.log was generated:

```
file = WildcardFileSearch( 'C:\Program Files\Cognos\TM1\Custom\
TM1Data\SData\tm1s*.log', 'tm1s20040211153827.log
');
```

This example begins searching the C:\Program Files\Cognos\TM1\Custom\ TM1Data\SData\ directory immediately after the tm1s20040211153827.log file, and returns the first subsequent file matching the 'tm1s*.log' wildcard sequence.

tm1s20040220175522.log is the first file that occurs after tm1s20040211153827.log and matches the wildcard sequence. Accordingly, the example returns tm1s20040220175522.log.

Chapter 7. TM1 TurboIntegrator Variables

The IBM Cognos TM1 TurboIntegrator variables are listed here by categories.

TurboIntegrator Local Variables

When you execute a TurboIntegrator process, a set of implicit local variables is generated. Local variables exist only in the context of the process in which they are used, and are not available outside of the process. Local variables are destroyed when a process exits. These variables, listed below, can be overwritten to manipulate a process.

DatasourceNameForServer

This TurboIntegrator local variable sets the name of the data source (.cma file, cube name, ODBC source) used by the server when executing the process.

Syntax

```
DatasourceNameForServer='Name';
```

Argument	Description
Name	For a .cma data source, the full path of the .cma file. For cubes, the cube name prefaced with the string 'local:'. For an ODBC source, the source name.

DatasourceNameForClient

This TurboIntegrator local variable sets the name of the data source (.cma file, cube name, ODBC source) used by the client when creating or editing the process.

Syntax

```
DatasourceNameForClient='Name';
```

Argument	Description
Name	For a .cma data source, the full path of the .cma file. For cubes, the cube name prefaced with the string 'local:'. For an ODBC source, the source name.

DatasourceType

This TurboIntegrator local variable sets the type of the data source.

Syntax

```
DataSourceType='Type';
```

Argument	Description
Type	Valid types include: 'CHARACTERDELIMITED', 'POSITIONDELIMITED', 'VIEW', 'SUBSET', ODBC' and 'OLEDBOLAP', 'SAPCHARACTERISTICTEXTS'

DatasourceUsername

This TurboIntegrator local variable sets the name used to connect to the data source.

Syntax

```
DatasourceUserName='Name';
```

Argument	Description
Name	The name used to connect to the data source set with DatasourceNameForServer.

DatasourcePassword

This TurboIntegrator local variable sets the password used to connect to the data source.

Syntax

```
DatasourcePassword='Password';
```

Argument	Description
Password	The password used to connect to the data source set with DatasourceNameForServer.

DatasourceQuery

This TurboIntegrator local variable sets the query string to use with the data source.

Syntax

```
DatasourceQuery='Query';
```

Argument	Description
Query	The query string to use with the data source that was set with DatasourceNameForServer.

DatasourceCubeview

This TurboIntegrator local variable sets the view to process if the DatasourceType is 'VIEW'.

Syntax

```
DatasourceCubeview='ViewName';
```

Argument	Description
ViewName	The name of the view to be processed. This must be an existing view of the cube specified by the DataSourceNameForServer variable.

DatasourceDimensionSubset

This TurboIntegrator local variable sets the subset to process if the DatasourceType is 'SUBSET.'

DatasourceNameForServer=*Dimension name* is also needed in conjunction with DATASOURCEDIMENSIONSUBSET so TM1 can identify where the subset is located.

Syntax

```
DatasourceDimensionSubset='SubsetName';
```

Argument	Description
SubsetName	The name of the subset to be processed.

DatasourceASCIIDelimiter

This TurboIntegrator local variable sets the ASCII character to be used as a field delimiter when the DatasourceType is 'CHARACTERDELIMITED''.

Syntax

```
DatasourceASCIIDelimiter='Character';
```

Argument	Description
Character	The ASCII character to be used as a delimiter.

DatasourceASCIIDecimalSeparator

This TurboIntegrator local variable sets the decimal separator to be used in any conversion of a string to a number or a number to a string. If you set this variable you must also set the DatasourceASCIIThousandSeparator variable.

Syntax

```
DatasourceASCIIDecimalSeparator='Character';
```

Argument	Description
Character	The ASCII character to be used as a separator.

DatasourceASCIIThousandSeparator

This TurboIntegrator local variable sets the thousands separator to be used in any conversion of a string to a number or a number to a string.

If you set this variable you must also set the DatasourceASCIIDecimalSeparator variable.

Syntax

```
DatasourceASCIIThousandSeparator='Character';
```

Argument	Description
Character	The ASCII character to be used as a separator.

DatasourceASCIIQuoteCharacter

This TurboIntegrator local variable sets the ASCII character used to enclose the fields of the source file when DatasourceType is 'CHARACTERDELIMITED'.

Syntax

```
DatasourceASCIIQuoteCharacter='Character';
```

Argument	Description
Character	The ASCII character that encloses fields in the data source.

DatasourceASCIIHeaderRecords

This TurboIntegrator local variable indicates the number of records to be skipped before processing the data source.

Syntax

```
DatasourceASCIIHeaderRecords=N;
```

Argument	Description
N	The number of records to be skipped before processing the data source.

Value_Is_String

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines whether the current cell should be treated as a string or a numeric value.

Syntax

```
Value_Is_String=N;
```

Argument	Description
N	Value indicating if the current cell is a string or a numeric value. 0 dictates that the cell is a number; anything else means the cell is treated as a string.

NValue

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines the value of the current cell when Value_Is_String is 0. (That is, when the current cell is numeric.)

Syntax

```
Nvalue=N;
```

Argument	Description
N	The value of the current cell.

SValue

When the DatasourceType is 'VIEW', this TurboIntegrator local variable determines the value of the current cell when Value_Is_String is not 0. (That is, when the current cell contains a string.)

Syntax

```
Svalue='String';
```

Argument	Description
String	The value of the current cell.

OnMinorErrorDoItemSkip

This TurboIntegrator local variable instructs TurboIntegrator to skip to the next record when a minor error is encountered while processing a record.

This variable is useful in scenarios where a single bad field/value in a record causes multiple minor errors.

For example, if you have 100 CELLPUTN functions in a process and one of the fields in a given record is 'bad' or invalid, the minor error count is incremented by 100. (1 for each CELLPUTN function that encounters the error.) These 100 minor errors count towards the minor error limit defined by MinorErrorLogMax. A TurboIntegrator process fails when it surpasses the number of minor errors defined by MinorErrorLogMax.

If OnMinorErrorDoItemSkip=1; is included in the Prolog tab of the process, the process immediately skips to the next record when a 'bad' or invalid field is encountered in a source record. Using the above example, this results in the minor error count being incremented by just 1, rather than 100.

Syntax

```
OnMinorErrorDoItemSkip=N;
```

Argument	Description
N	Value indicating if item should be skipped when a minor error is encountered. 1 (or any other non-zero value) dictates that the process should skip to the next record when a minor error is encountered. 0 indicates that TurboIntegrator should continue processing the current record when a minor error occurs.

MinorErrorLogMax

This TurboIntegrator local variable defines the number of minor errors that will be written to the TM1ProcessError.log file during process execution. If this variable is not defined in the process, the default number of minor errors written to the log file is 1000.

Syntax

```
MinorErrorLogMax=N;
```

Argument	Description
N	Value indicating the number of errors that should be written to the log file. Specify an integer greater than zero to set the maximum number of errors written to the log file. Specify a value of 0 to log no errors during process execution. Specify a value of -1 to allow an unlimited number of minor errors to be written to the log file.

```
The following table provides an example error log message
and the corresponding result.
```

Example	Result
MinorErrorLogMax=750;	The log file will accept up to 750 errors.
MinorErrorLogMax=0;	No errors will be written to the log file.
MinorErrorLogMax=-1;	No limit on the number of errors written to the log file.

DataSourceODBOCatalog

This TurboIntegrator local variable sets the name of the database collection that contains the cubes, dimensions or other objects to which you want to connect. For Microsoft Analysis Services, this is the name of the database.

Syntax

```
DataSourceODBOCatalog='Catalog';
```

Argument	Description
Catalog	The name of the database collection to which you want to connect.

DataSourceODBOConnectionString

This TurboIntegrator local variable sets any additional connection parameters that may be required to connect to the OLAP server.

Syntax

```
DataSourceODBOConnectionString='String';
```

Argument	Description
String	The value used to define additional connection parameters. Assign these parameters to this variable, delimited by semi-colons.

DataSourceODBOCubeName

This TurboIntegrator local variable sets the name of the cube from the OLAP server that you want to use as a data source.

Syntax

```
DataSourceODBOCubeName='Name';
```

Argument	Description
Name	The name of the cube to be used.

DataSourceODBOHierarchyName

This TurboIntegrator local variable sets the name of the hierarchy for the specific dimension you are using as a data source. You use this variable for other OLAP products, such as SAP BW, where a hierarchy is a separate object.

This variable is not used with TM1 data sources.

Syntax

```
DataSourceODBOHierarchyName='Name';
```

Argument	Description
Name	The name of the hierarchy for a specific dimension.

DataSourceODBOLocation

This TurboIntegrator local variable sets the name of the location (system) where the OLAP server is running.

TM1 uses this variable, but other OLAP servers do not. For TM1 , this is the location where the Admin Host is running.

Syntax

```
DataSourceODBOLocation='Location';
```

Argument	Description
Location	The name of the location (system) for the OLAP server.

DataSourceODBOProvider

This IBM Cognos TM1 TurboIntegrator local variable sets the name of the ODBO provider that you want to use as a data source. This is the full name that is assigned by the ODBO provider manufacturer to identify their multidimensional database server.

You must use the name of an ODBO provider that is installed on your server.

Syntax

```
DataSourceODBOProvider='Provider';
```

Argument	Description
Provider	The name of the ODBO provider to use as a data source. Commonly-used provider names include: TM1 OLE DB MD Provider Microsoft OLE DB Provider for OLAP Services 8.0 SAP BW OLE DB Provider

DataSourceODBOSAPClientID

This TurboIntegrator local variable sets the client number that corresponds to the UI version on the SAP server to which you want to connect.

Syntax

```
DataSourceODBOSAPClientID='ID';
```

Argument	Description
ID	A number that corresponds to the UI version on the SAP server. For example, 498.

DataSourceODBOSAPClientLanguage

This TurboIntegrator local variable sets the language specification for the language of the SAP system to which you want to connect.

Syntax

```
DataSourceODBOSAPClientLanguage='Language';
```

Argument	Description
Language	The language specification of the SAP system. For US English, use EN. For German, use DE. For other languages, refer to your SAP documentation.

TurboIntegrator Global Variables

This type of TurboIntegrator variable is associated with an individual TM1 chore or with an individual process and any attendant sub-processes. There are two types of global variables: implicit and user-defined. Implicit global variables are described here. User-defined global variables are described below.

Global variables can be used in two ways:

- Global variables can be declared within a process that is part of a given chore. Once declared, the global variables are available to all other processes that are part of the chore. The variables persist while the chore is executing and for the duration of the current server session. Global variables are destroyed upon server shutdown.
- Global variables can be declared in one process and be made available to any subsequent processes called by the ExecuteProcess( ) function. These sub-processes must use the same global variable declaration statements (described below) to access the global variables.

In the event that a global variable name is identical to a local variable name, the local variable definition takes precedence and overrides the global variable.

Global variables are declared in a TurboIntegrator process using one of the following two functions:

- NumericGlobalVariable('VariableName');.
- StringGlobalVariable('VariableName');.

NumericGlobalVariable('VariableName');

Use this function to declare a numeric global variable.

StringGlobalVariable('VariableName');

Use this function to define a string global variable.

Implicit Global Variables

When you execute a TurboIntegrator process, a set of implicit global variables is generated. If the process generating the variables is part of a chore, these global variables are available to and can be shared by all other processes within the chore. In addition, all implicit global variables in a process are available to and can be shared by any subsequent processes called by the ExecuteProcess( ) function.

Though implicit variables are generated by the TurboIntegrator process, you must declare a variable before it can be used in a process

Implicit global variables are declared in a TurboIntegrator process using the NumericGlobalVariable('VariableName');:

Click a link below for details on a specific implicit global variables.

- DataMinorErrorCount.
- MetadataMinorErrorCount.
- ProcessReturnCode.
- PrologMinorErrorCount.

For example, to use the PrologMinorErrorCount implicit global variable in a process, you must first declare the variable as follows:

```
NumericGlobalVariable('PrologMinorErrorCount');
```

DataMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Data portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
DataMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Data portion of the process.

MetadataMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Metadata portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
MetadataMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Metadata portion of the process.

ProcessReturnCode

This TurboIntegrator global variable stores the exit status of the most recently executed TurboIntegrator process.

Syntax

```
ProcessReturnCode=StatusCode;
```

Status Code	Description
ProcessExitByChoreQuit()	indicates that the process exited due to execution of the ChoreQuit function
ProcessExitNormal()	indicates that the process executed normally
ProcessExitMinorError()	indicates that the process executed successfully but encountered minor errors
ProcessExitByQuit()	indicates that the process exited because of an explicit "quit" command

Status Code	Description
ProcessExitWithMessage()	indicates that the process exited normally, with a message written to `tm1server.log`.
ProcessExitSeriousError()	indicates that the process exited because of a serious error
ProcessExitOnInit()	indicates that the process aborted during initialization
ProcessExitByBreak()	indicates that the process exited because it encountered a ProcessBreak function

PrologMinorErrorCount

This TurboIntegrator global variable counts the minor errors that occur in the Prolog portion of a TurboIntegrator process. For each minor error encountered, the variable value is incremented by 1.

Syntax

```
PrologMinorErrorCount=N;
```

Argument	Description
N	The number of minor errors encountered in the Prolog.

TurboIntegrator User Variables

This type of variable is associated with an individual TM1 user, not with any particular process or chore. User variables can be manipulated from within any TurboIntegrator process or chore while the user with which the variable is associated is logged on to the server.

User variables must be explicitly declared. Once declared, user variables persist for the life of the user's TM1 session (until the user logs off or is disconnected from the server).

User variables are declared in a TurboIntegrator process using one of the following two functions:

- NumericSessionVariable('VariableName');.
- StringSessionVariable('VariableName');.

User variables are created the first time such a declaration is encountered in any running TurboIntegrator process.

Once created, the variable name may be referenced and used just like any local or global variable, expect that the variable value persists across processes and chores only for as long as the user who created the variable is logged on to the server.

NumericSessionVariable('VariableName');
Use this function to declare a numeric user variable.

StringSessionVariable('VariableName');

Use this function to define a string user variable.

Chapter 8. MDX Function Support

All TM1 supported Microsoft-defined and TM1 -specific functions are listed in this section.

Support for Microsoft-defined MDX Expressions and Functions

TM1 supports Microsoft-defined MDX expressions and functions.

The TM1 implementation of these functions and expressions is based on the definitions in the Microsoft MSDN library, which is available at the Microsoft MSDN website.

List of Supported Member Expressions

The following MDX member expressions are supported.

- <dimension>.CURRENTMEMBER
- <member>.FIRSTCHILD
- <member>.FIRSTSIBLING
- <member>.LAG
- <member>.LASTCHILD
- <member>.LASTSIBLING
- <member>.LEAD
- <member>.NEXTMEMBER
- <member>.PARENT
- <member>.PREVMEMBER

List of Supported Member Functions

The following MDX member functions are supported.

- ANCESTOR(...)
- COUSIN(...)
- OPENINGPERIOD(...)
- PARALLELPERIOD(...)

List of Supported Numeric Functions

The following MDX numeric functions are supported.

- AGGREGATE(...)
- AVG(...)
- CORRELATION(...)
- COUNT(...)
- COVARIANCE(...)
- LINREGINTERCEPT(...)
- LINREGPOINT(...)
- LINREGR2(...)
- LINREGSLOPE(...)
- LINREGVARIANCE(...)

- MAX(...)
- MEDIAN(...)
- MIN(...)
- RANK(...)
- STDDEV(...)
- SUM(...)
- VAR(...)

List of Supported Set Expressions

The following MDX set expressions are supported.

- <dimension>.MEMBERS
- <level>.MEMBERS
- <member>. CHILDREN
- <member>.SIBLINGS

List of Supported Set Functions

The following MDX set functions are supported.

- ADDCALCULATEDMEMBERS(...)
- BOTTOMCOUNT(...)
- BOTTOMPERCENT(...)
- BOTTOMSUM(...)
- CROSSJOIN(...)
- DESCENDANTS(...)
- DISTINCT(...)
- DRILLDOWNLEVEL(...)
- DRILLDOWNLEVELBOTTOM(...)
- DRILLDOWNLEVELTOP(...)
- DRILLDOWNMEMBER(...)
- DRILLDOWNMEMBERBOTTOM(...)
- DRILLDOWNMEMBERTOP(...)
- DRILLUPMEMBER(...)
- DRILLUPLEVEL(...)
- EXCEPT(...)
- EXTRACT(...)
- FILTER(...)
- GENERATE(...)
- HEAD(...)
- HIERARCHIZE(...)
- INTERSECT(...)
- LASTPERIODS(...)
- ORDER(...)
- PERIODSTODATE(...)
- TOPCOUNT(...)
- TOGGLEDRILLSTATE(...)
- TOPPERCENT(...)

- TOPSUM(...)
- SUBSET(...)
- UNION(...)

List of Supported Tuple Expressions

The following MDX tuple expressions are supported.

- <set>.CURRENTMEMBER
- <set>[.ITEM](...)

TM1 -Specific MDX functions

TM1 supports several TM1 -specific MDX expressions. You can apply these expressions while developing MDX applications to run against the server or when creating/editing dynamic subsets in the Expression Window of the Subset Editor.

TM1FILTERBYPATTERN(<set>, <pattern_str>)

This TM1 -specific MDX function returns all the members in <*set*> with names matching the pattern <*pattern_str*>.

The syntax of <*pattern_str*> is the same used for the Select By Regular Expression option on the Subset Editor.

TM1FILTERBYLEVEL(<set>, <level_number>)

This TM1 -specific MDX function returns all the members in <*set*> of the specified <*level_number*>.

<*level_number*> is a number specifying the TM1 level number not an MDX level number.

TM1DRILLDOWNMEMBER(<set1>, <set2>|ALL [,RECURSIVE])

This TM1 -specific MDX function is similar to the DRILLDOWNMEMBER function from Microsoft , but it has been adjusted to match the functionality of the Expand button {bmct expand_button.bmp} on the Subset Editor.

ALL means drilldown all the members in <*set1*>.

RECURSIVE means that when one member from <*set1*> is being drilled down upon, every consolidated member resulting from that expansion will also be recursively drilled down until level 0 (TM1 level 0) is reached.

TM1Member

This function returns a member from a specified tuple.

A null member reference is returned when any of the following conditions are encountered:

- A null Tuple parameter
- An out-of-range numeric Index parameter
- A dimension or hierarchy parameter not found in the passed tuple.

Syntax

```
TM1Member(Tuple, MemberSpecifier);
```

Argument	Description
Tuple	An expression that resolves to a tuple.
MemberSpecifier	This parameter can be either a 0-based numeric index into the tuple or the name of a dimension/hierarchy associated with the tuple. See below for examples showing both parameter types.

Example

TM1Member ([model].Members.Item(23),0)] This example uses a numeric index into the tuple as the MemberSpecifier argument.

TM1Member([model].Members.Item(23), [Model])] This example uses the name of a dimension associated with the tuple as the MemberSpecifier argument.

TM1SORT(<set>, ASC|DESC)

This TM1 -specific MDX function sorts <*set*> alphabetically.

ASC sorts A-Z

DESC sorts Z-A

TM1SORTBYINDEX(<set>, ASC|DESC)

This TM1 -specific MDX function sorts <*set*> by the index value of the members.

ASC sorts by ascending index value.

DESC sorts by descending index value.

TM1SUBSETALL(<dimname>)

This TM1 -specific MDX function returns the TM1 subset All of <*dimname*>.

TM1SubsetToSet

This function returns the members of a TM1 subset.

TM1SubsetToSet is equivalent to the <*dimension*>.<*subsetname*> expression, but does not require string literals. Instead, TM1SubsetToSet lets you use expressions that resolve to the appropriate dimension and subset.

If identically-named dimensions are found in two different cubes, enter a fully-qualified name such as [cube].[dimension].

Syntax

```
TM1SubsetToSet(Cube.Dimension_exp, Subset_exp);
```

Argument	Description
Dimension_exp	An expression that resolves to a valid TM1 dimension name. Use the fully-qualified format of [cube].[dimension] to identify the dimensions if there are identically-named dimensions in the cubes.
Subset_exp	An expression that resolves to a valid subset of the dimension returned by Dimension_exp. When resolving an expression for a subset, the server searches first in the private subset list and then in the public list.

Example

MDX sample code with dimension name `[Corp Planning Hry]` in the last line:

```
{INTERSECT(EXCEPT(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),
TM1FILTERBYLEVEL(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),0)),{
TM1SubsetToSet([Corp Planning Hry],"elist")}),[Corp Planning Hry].[FixedAssets]}
```

MDX sample code with fully-qualified dimension name `[Corp Planning Hry].[Corp Planning Hry]` and `[Corp Planning Hry].[FixedAssets]` in the last line:

```
{INTERSECT(EXCEPT(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),
TM1FILTERBYLEVEL(DESCENDANTS([Corp Planning Hry].[Fixed Assets]),0)),{
TM1SubsetToSet([Corp Planning Hry].[Corp Planning Hry],"elist")}),
[Corp Planning Hry].[FixedAssets]}
```

TM1TupleSize

This function returns the number of members in a tuple.

Syntax

```
TM1TupleSize(Tuple);
```

Argument	Description
Tuple	An expression that resolves to a tuple. The function returns 0 if the Tuple argument does not resolve to a valid tuple, or of the tuple is null or empty.

TM1 -Specific MDX expressions

TM1 supports several TM1 -specific MDX expressions. You can apply these expressions while developing MDX applications to run against the server or when creating/editing dynamic subsets in the Expression Window of the Subset Editor.

<dimension>.<subsetname>

This TM1 -specific MDX expression returns members of <subsetname> in <dimension>.

Since the same syntax (<dimension>.IDENTIFIER)is used for members and levels, a subset with the same name of a member or a level will never be instantiated.

When searching for a subset, the server searches first in the private subset list and then in the public list.

<member>.ANCESTORS

This TM1 -specific MDX expression returns the ancestors of <member>.

For example, assuming the following hierarchy of the Month dimension:

- Year
  - 1 Quarter
  - Jan
  - Feb
  - Mar

the expression

month.jan.ANCESTORS

returns the set

{ 1Quarter, Year }.

If the member has more than one immediate parent, the expression returns the set containing the first parent in the default hierarchy. Consider a hierarchy of a Region dimension, where the member Belgium has more than one immediate parent, being Benelux and Europe. In this case, the expression

region.belgium.ANCESTORS

returns the set

{ Benelux, Europe }.

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information 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.

Trademarks

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 at " Copyright and trademark information " at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- 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.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft product screen shot(s) used with permission from Microsoft.

Index