IBM Planning Analytics
Version 2 Release 0

TM1 TurboIntegrator

Note

Before you use this information and the product it supports, read the information in "Notices" on page 93.

Product Information

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

Licensed Materials - Property of IBM

Last updated: 2017-09-19

Contents

Introduction

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

This manual describes how to use IBM Cognos TM1 TurboIntegrator to import data and metadata from a variety of Business Analytics sources.

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

Samples disclaimer

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

Accessibility features

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

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

Forward-looking statements

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

Security considerations

For security considerations for IBM Planning Analytics, see *Planning Analytics Installation and Configuration*. Information on managing user and group authentication can be found in the *Managing Users and Groups* chapter of the *TM1 Operations* documentation.

Chapter 1. TurboIntegrator Basics

This section describes basic information about how to import data into an IBM Cognos TM1 cube using TurboIntegrator.

TurboIntegrator lets you design a process that recognizes the data structure of the source and transforms it into the appropriate structure TM1 needs. Once the TI process is designed, you can re-run it or schedule it to be used when importing data from a dynamic source. Subsequent sections describe the steps used to import data from specific types of sources.

Before you begin using TurboIntegrator, be sure you understand the information that applies to all kinds of sources described in this chapter.

Data Sources Available with TurboIntegrator

Using TM1 TurboIntegrator, you can import data from these data sources:

- Comma-delimited text files including ASCII files.
- Relational database tables that are accessible through an ODBC data source.
- Other cubes and views.
- Microsoft Analysis Services.
- SAP via RFC.
- IBM Cognos packages

See the other sections in this manual for details about each of these source types.

String Length Limit in TurboIntegrator

TurboIntegrator is capable of handling string data in sizes of up to 8000 single-byte characters at a time. This limit applies when your TI process is performing actions such as assigning a value to a variable or importing individual records of data. Any value or record longer than 8000 single-byte characters is truncated.

This limit applies when your TI process is performing actions such as assigning a value to a variable or importing individual records of data. Any value or record longer than 8000 single-byte characters is truncated.

For example, if you are importing lines of data from a text file, each line of text can be no more than 8000 characters long. If you are importing data from a comma delimited file, each record in the file can be no more than 8000 characters long.

Importing Options

When you import data using TurboIntegrator, you have the following options:

- Create a cube and populate it with data imported from the source.
- Re-create a cube. This destroys a currently-existing cube and recreates it, allowing you to change both data and metadata during the import.
- Update an existing cube, maintaining the cube's structure. This enables you to import data into an existing cube structure.
- Create a dimension from data imported from a source.
- Update a dimension from imported data.

You can use TurboIntegrator to perform any combination of these actions.

TurboIntegrator Functions

TurboIntegrator includes a series of functions that let you manipulate cubes, views, dimensions, elements, and other TM1 objects while importing data.

In addition to these TurboIntegrator functions, you can also incorporate all standard TM1 rules functions in a TurboIntegrator process, with the exception of the STET function.

TurboIntegrator functions are described in "TM1 TurboIntegrator Functions" in the IBM Cognos TM1 *Reference*.

Processes and Chores

You import data with TurboIntegrator by defining a *process*. A process is the TM1 object that consists of:

- A description of a data source.
- A set of variables corresponding to each column in the data source.
- A set of maps that define relationships between variables and data structures in the TM1 database.
- A Prolog procedure, consisting of a series of actions to be executed before the data source is processed.
- A Metadata procedure, consisting of a series of actions that update or create cubes, dimensions, and other metadata structures.
- A Data procedure, consisting of a series of actions that update or transform data in your TM1 database.
- An Epilog procedure to be executed after the data source is processed.
- A set of parameters that can be used to generalize a process so it can be used in multiple situations.

Chores are a container object for a set of TM1 processes. Chores allow you to run processes in a certain order and schedule processes to be run at a certain time. For details, see Chapter 8, “Scheduling a Process for Automatic Execution with Chores,” on page 49.

Order of Operations within a TurboIntegrator Process

A TurboIntegrator process includes several procedures: Prolog, Metadata, Data, and Epilog. These procedures can be viewed as sub-tabs of the Advanced tab in the TurboIntegrator editor.

When you define a data source, set variables, and specify data actions for a process, TM1 generates scripts that are executed when you run the TurboIntegrator process. These scripts are placed in the appropriate procedure sub-tab of the TurboIntegrator editor. You can also create your own scripts in any of the procedure sub-tabs using TurboIntegrator functions and rules functions.

When you run a TurboIntegrator process, the procedures are executed in the following sequence:

1. The Prolog procedure is executed *before* the data source for the TurboIntegrator process is opened.
2. 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.
3. If the data source is anything other than None, TurboIntegrator opens the data source for the process.
4. All lines in the Metadata procedure are sequentially executed against the first record in the data source. All lines are then sequentially executed against the second record in the data source, and so on until all records are processed.
5. All lines in the Data procedure are sequentially executed against the first record in the data source. All lines are then sequentially executed against the second record in the data source, and so on until all records are processed.
6. TurboIntegrator closes the data source after the Data procedure is completed.
7. The Epilog procedure is executed.
8. TM1 closes the TurboIntegrator process.

Notes on TurboIntegrator Processes

Keep in mind the following items when creating and editing TurboIntegrator processes.

- TurboIntegrator compiles a new or altered dimension only at the conclusion of the procedure in which the dimension is created or altered.

  In the case of a new dimension, this means that you cannot access the new dimension (through TurboIntegrator or otherwise) until the procedure in which the dimension is created has finished processing all records in the data source. In the case of an altered dimension, this means that you cannot access any new elements in the dimension until the procedure in which the dimension is altered has finished processing.
- TurboIntegrator and rules functions (with the exception of STET) can be used in any procedure of a process. Further, there are no restrictions regarding which functions can be used in a given procedure; all functions are valid in any TurboIntegrator procedure.
- See "Formulas" in the Introduction to Rules chapter of the IBM Cognos TM1 *Rules* documentation for information on using different kinds of operators, such as logical and arithmetic operators in TI processes and rules.
- In the TurboIntegrator process, null values are converted into zeroes for Numeric values and NULL values are converted into empty strings for String Values.
- If you try to put a Consolidated element under an existing N-level element, the N-level element changes to a Consolidated element and any data in the original N-level element will be lost.

You must, however, create a logical sequence of functions to ensure that a process accomplishes your goal. For example, if you want to build a process that adds new elements to a dimension and updates data values for the new elements, you must be sure that the process adds the new elements and compiles the dimension *before* attempting to update data values for the new elements. In most circumstances, you would add the new elements in the Metadata procedure using the DimensionElementInsert function, then update values in the Data procedure using the CellPutN function.

Using the above example, if you attempt to build a process in which both new elements are added *and* corresponding data values are updated in the Data procedure, the process will fail. The failure occurs because, as noted above, altered dimensions are compiled only at the conclusion of a procedure. Until the dimension is compiled, the new elements do not exist. TurboIntegrator cannot update data values for elements that do not exist, so the process fails.

Simultaneous Connections to the Same TM1 Server

Do not perform any operation within a TurboIntegrator process that creates a new connection (logs in) to the *same* TM1 server where the process is already running. This type of scenario could cause a deadlock situation between the two logins or threads causing the server to hang or possibly crash.

For example, avoid the following scenarios:

- Do not use a TI process to launch an ODBO MDX query (via theTM1 OLE DB MD Provider) into the *same* server. This scenario can result in both the process and the query waiting on each other to finish.
- Do not use the TI function, `ExecuteCommand`, to call out of a TI process to run *and wait* (Wait argument set to 1) for an external program that logs back into the *same* server. This includes any custom application or any IBM Cognos application, such as the TM1 ETLDAP utility, that could possibly connect back to the same server.

Be aware that using the `ExecuteCommand` function with its Wait argument set to 1 risks hanging the server *even if* the external program *doesn't* log back into the same server. If the external program encounters its own problem and becomes hung, the TI process essentially hangs waiting for the external program to finish executing.

Aliases in TurboIntegrator Functions

An alias name can be used in place of the corresponding element principal name in rules or in TurboIntegrator functions.

Using Personal Workspaces and Sandboxes with TurboIntegrator Processes

This section describes how to use Personal Workspaces sandboxes with TurboIntegrator processes and functions.

Manually Running a TurboIntegrator Process with a Personal Workspace or Sandbox

You can manually run a process with the currently active sandbox in Server Explorer by selecting the **Use Active Sandbox** property for that process.

The active sandbox is determined by which sandbox is currently selected in the Cube Viewer. For Personal Workspaces the only available sandbox is the [Default].

Note: Chores, and the processes they contain, cannot run against a Personal Workspace or sandbox. When a process runs as part of chore, it can only run against base data.

Procedure

1. In Server Explorer, open a view in the Cube Viewer.
2. Click the sandbox in the list of available sandboxes to select the sandbox you want to use with the process.
3. In the Tree pane, right-click the process and click **Use Active Sandbox** to enable the option.
4. Right-click the process and click **Run**.

Results

The process runs using the current active sandbox.

Using TurboIntegrator Functions with Sandboxes

The following TurboIntegrator functions allow a TurboIntegrator process to interact with Personal Workspaces and sandboxes.

- GetUseActiveSandboxProperty
- SetUseActiveSandboxProperty
- ServerActiveSandboxGet
- ServerActiveSandboxSet

These functions are similar to the **Use Active Sandbox** property available in the Server Explorer interface.

For more information, see the section about TurboIntegrator sandbox functions in the IBM Cognos TM1 *Reference*.

Chapter 2. Importing a Text File

This section describes how to import comma-delimited text data, such as ASCII, with IBM Cognos TM1 TurboIntegrator. Though each TM1 process is unique, and importing from other types of data sources varies slightly, this section describes the steps common to most processes. The procedures and examples use the file NewEngland.cma, which is installed as part of the sample data that is included with TM1 .

Creating a Dimension from a Text File

You can use TurboIntegrator to create a dimension from a list of element names in a data source. This is the fastest way to create a dimension containing hundreds or thousands of elements.

When you create a dimension with TurboIntegrator, you define a process that is saved as an object on your TM1 server. This process can be accessed by other users, and can be executed on demand or at a scheduled interval.

To create a dimension with TurboIntegrator:

1. Define the data source to TM1 . See "Defining a Data Source" on page 5.
2. Identify the variables TM1 will encounter. See "Identifying Variables in the Data Source" on page 6.
3. Map the variables to their data types. See "Mapping Variables" on page 8.
4. Save the process and run it. See "Saving and Executing the TurboIntegrator Process" on page 10.

Defining a Data Source

Whenever you use TurboIntegrator, the first step requires you to define the data source from which you will be reading data. This example defines the ASCII file called NewEngland.cma as the data source for this TurboIntegrator process.

Procedure

1. Right-click **Processes** in the left pane of the Server Explorer and choose **Processes**, **Create New Process**.
2. Click **Text** on the Data Source tab.

   The TurboIntegrator window opens.
3. Click **Browse**.

   The Select Input File dialog box opens.
4. Navigate to NewEngland.cma, select it, and click **Open**.

   NewEngland.cma is available in either the PData or SData sample data directory. If you accepted the default installation directory for TM1 , the full path to the file is

   ```
   C:\Program Files\Cognos\TM1\Custom\TM1Data\SData\NewEngland.cma
   ```

   or

   ```
   C:\Program Files\Cognos\TM1\Custom\TM1Data\PData\NewEngland.cma.
   ```

   You may receive a message indicating that you should use Universal Naming Convention (UNC) to specify the file location. If you are going to consistently run your process against an ASCII file, you should use UNC and ensure the following:

   - If running a Microsoft Windows TM1 server, the ASCII file should reside on a shared Windows directory so that the server can access it.
   - If running a TM1 server on a UNIX operating system, your file should reside on a shared network directory, one which both the TM1 Windows client and the TM1 UNIX server can see.

   Note: If running aTM1 server on a UNIX operating system, the input source filename *cannot* contain any upper-case characters or spaces.

5. Click **OK** on the warning box.
6. Complete the TurboIntegrator dialog box as follows:

   NewEngland.cma is a delimited source that uses commas as its delimiter; double quote as a quote character; no title records; a period as a decimal separator; and a comma as a thousands separator.

   To define this source enter the following settings:

   - At Delimiter Type, select **Delimited**.
   - Select **Comma** as the Delimiter.
   - Enter **"** at Quote Char.
   - Leave the Number of Title Records field blank.
   - Enter **.** at Decimal Separator.
   - Enter **,** at Thousands Separator.
7. Click **Preview**.

   TurboIntegrator shows you a sample of your source data at the bottom of the window.

Using Fixed Length Records
TurboIntegrator can also import data from text files that use fixed width fields. To specify that the data source has fixed width fields, after specifying the location of your data source file select the Delimiter Type of **Fixed Width**, then click **Set Field Width**.

The Data Preview dialog box displays the first three records of your source data. To set the field widths based on the contents of records in your data source:

Procedure

1. Click the **1** column heading.

   A break line displays in the column heading and extends through the three records.
2. Click the break line and drag it to a position that separates the first column from the second column.

   A new column heading (2) displays.
3. Click column heading **2** and drag the new break line to a position that separates the second column from the third column.
4. Set break lines for all remaining columns in the text source.
5. Click **OK** to return to the TurboIntegrator window.

Identifying Variables in the Data Source

After you define a data source, TurboIntegrator assigns a variable to each column in the source. You must identify these variables by type and content.

To illustrate this process, consider the following text data:

Table 1: sample text data

Column 1	**Column 2**	**Column 3**	**Column 4**	**Column 5**	**Column 6**
New England	Massachusetts	Boston	Supermart	Feb	2000000
New England	Massachusetts	Springfield	Supermart	Feb	1400000
New England	Massachusetts	Worcester	Supermart	Feb	2200000
New England	Connecticut	Hartford	Supermart	Feb	1240000
New England	Connecticut	New Haven	Supermart	Feb	2700000
New England	Connecticut	Greenwich	Supermart	Feb	1700000

The first 3 columns form a hierarchy for a Location dimension that you will build from the source text file:

- The New England consolidation is at the top of the hierarchy.
- The states Massachusetts and Connecticut are one level below New England.
- The third column, containing city names such as Boston and Hartford, supplies simple elements at the lowest level of the hierarchy.
- The remaining columns are not used for the creation of the Location dimension.

Here is the Variables tab on the TurboIntegrator window for this data structure:

Table 2: Variables tab

Variable Name	**Variable Type**	**Sample Value**
V1	String	New England
V1	String	New England
Massachusetts	String	Massachusetts
Boston	String	Boston
SuperMart	String	SuperMart
Feb	String	Feb
V6	Numeric	2000000

TurboIntegrator assigns a variable name to each column, and assigns a variable type based on the sample value for each column.

The default variables names, such as V1 and Massachusetts, can be changed. It is good practice to give the variables a meaningful name. Having meaningful names makes the TurboIntegrator scripts easier to read and troubleshoot.

To edit a variable name, click the name in the Variable Name column and type a new name. For this exercise, the names of the first three variables were edited like this:

Sample Value	**Variable Name**
New England	Region
Massachusetts	State
Boston	City

A variable name must begin with a letter, and can contain only these characters:

Character	**Description**
Upper-case Letters	A through Z
Lower-Case Letters	a through z
Digits	0 through 9
Period	.
Underscore	_
Dollar Sign	$

The Variable Type field identifies the contents of the column. For instance, the first column of this data contains the string "New England". TurboIntegrator correctly identifies the variable type as String.

Note: The Variable Type fields are usually set accurately for ASCII data, but not for data extracted from an ODBC data source.

The Contents field can be defined with one of these settings:

Option	**Description**
Ignore	Ignore the contents of the column when processing the data source.
Element	The column contains simple elements for the dimension you want to create.
Consolidation	The column contains consolidated elements for the dimension you want to create.
Data	The column contains data values. For this example, you should ignore the column containing the data values. Columns containing data values are not imported when you are creating a dimension.
Attribute	The column contains element attributes for the dimension you want to create.
Other	The column contains data that does not fall into any of the previous four categories. Typically, this setting is used for columns containing data that will be processed through custom variables and formulas.

The text data in this example contains elements and consolidations for a Location dimension:

- It does not contain any attributes.
- It does contain data values, but those values are irrelevant to the creation of the Location dimension, as are the elements from other dimensions.

To define the variables for the Location dimension:

Procedure

1. Click the **Variables** tab on the TurboIntegrator window.
2. Set the **Contents** field for variables Region, State, and City as specified here:

Variable	**Content**
Region	Consolidation
State	Consolidation
City	Element

- The variable Region is now identified as a consolidation
- The variable State is also identified as a consolidation
- The variable City is identified as a leaf level (non-consolidated) element.

Mapping Variables

After you identify the variables in your data source, you must map those variables to elements and consolidations.

To begin mapping variables, click the **Maps** tab on the TurboIntegrator window.

The Maps tab contains several additional tabs. The Cube tab is always available. All other tabs become available based upon the column contents you set on the Variables tab. For example, if you identify a column as containing elements, the Dimension tab becomes available. If you identify a column as containing consolidations, the Consolidations tab becomes available, and so on.

Disable Cube Mapping
You should not perform any cube actions when creating a dimension. To prevent cube mapping:

Procedure

1. Click the **Cube** tab.
2. Select **No Action** in the Cube Action box.

Mapping Dimensions
If you identify any columns in your data source as containing elements, you must map those elements to the dimension you are creating:

Procedure

1. Click the **Dimensions** tab.
2. Type **Location** in the Dimension field.

   If multiple elements map to the same dimension, type the dimension name for each element.

   When you type a new dimension name in the Dimension column, the Action column defaults to Create.

   If you type the name of an existing dimension, you have the option of recreating or updating the dimension. If you choose the Recreate action, the elements in the existing dimension are deleted and replaced by the data in the data source. If you choose the Update action, the dimension is updated with any new elements contained in the data source.
3. Select a type for each element from the appropriate Element Type menu. The element type indicates the type of data identified by the element variable. In TM1 , this setting is almost always Numeric.
4. Select an **Element Order** option. The element order determines how elements are added to the dimension during processing.

   The data in this example contains a single numeric element that maps to a new Location dimension. The completed Dimensions tab displays as follows.

Disable Data Mapping
You should not perform any data mapping when creating a dimension.

Example

Since you specified No Action on the Cube Mapping tab "Disable Cube Mapping" on page 9, this Data tab becomes unavailable.

Mapping Consolidations
If you identify any columns in your data source as containing consolidations, you must map consolidation paths for the dimension you are creating:

Procedure

1. Click the **Consolidations** tab.

   The tab displays the variables that are defined as consolidations, Region and State.

   You can define the consolidation hierarchy of the dimension by specifying the child variable of each consolidation variable.
2. The immediate child of the Region consolidation variable is State. Click the right angle bracket button in the Child Variable field for the Region consolidation, select **State**, and click **OK**.
3. The immediate child of the State consolidation variable is City. Click the right angle bracket button in the Child variable field for the State consolidation, select **City**, and click **OK**.
4. For each consolidation, click the **Component Order** button. The Component Element Ordering dialog box opens.
5. Click **Automatic**, **Name**, and **Ascending**.

   Note: When you are setting up multiple consolidations within the same dimension, all the consolidations must be set to the same Component Element Ordering settings. If you set two consolidations in the same dimension to different Component Element Ordering settings, TurboIntegrator produces a Sorting Information Mismatch error when you try to save and execute the process.

Saving and Executing the TurboIntegrator Process

After you define a data source and set up variables, the TurboIntegrator process is compiled and saved. To create the dimension, you execute the completed process.

Procedure

1. Click **File**, **Save** from the TurboIntegrator menu bar.

   The Save Process As dialog box opens.
2. Enter a name for the process and click **Save**.

   If TM1 encounters an error during compilation and saving, an error message indicates the nature of the error. The TurboIntegrator window remains active, so you can immediately correct any errors.

   TM1 saves the process as a server object under Processes in the Server Explorer. The process is now available for execution or modification.

   To execute the process and create the dimension, click **File**, **Execute** from the TurboIntegrator menu bar. You can also execute a process directly from the Server Explorer by selecting the process and clicking **Process**, **Execute Process.**

   If the process executes successfully, TM1 issues a confirmation message.

   If TM1 is unable to execute the process, a dialog box details the errors encountered during execution.

   When NewEngland.cma is processed, a new Location dimension is created.

Creating a Cube from a Text File

TurboIntegrator can also create an entire cube from a text file. This procedure also builds some dimensions and elements, and performs some data manipulation.

The procedure for building a cube is similar to the process for building a dimension:

1. Define the data source to TM1 . See "Defining the Cube Data Source" on page 10.
2. Identify the variables TM1 will encounter. See "Defining Cube Variables" on page 11.
3. Map the different variables to their different data types in the resulting cube. See "Mapping Cube Element Variables to Dimensions" on page 11, "Mapping Cube Data Variables" on page 12,"Mapping Cube Variables" on page 11, and "Mapping Consolidation Variables" on page 12.
4. Save the process and run it. See "Saving and Executing the Cube Process" on page 12.

TM1 includes a sample data directory called TI_data. TI_data contains a file called import_cube.csv. This example describes how to build a cube from import_cube.csv.

Defining the Cube Data Source

The first step in creating a cube from a text file is to define the data source.

Procedure

1. In the left pane of the Server Explorer, right-click the **Processes** icon and select **Create New Process**.
2. Click the **Data Source** tab on the TurboIntegrator window.
3. Choose **Text** as the Data Source Type.
4. Click the **Browse** button next to the Data Source Name field and select the **import_cube.csv** file in your TI_data directory. If you accepted the default installation directory, the full path to the TI_data directory is

   ```
   C:\Program Files\Cognos\TM1\Custom\TM1Data\TI_Data
   ```
5. Set the Delimiter Type to **Delimited**, and choose **Comma** as the Delimiter.

Ignore the Quote Char and Number of Title Records fields for this example.

6. Make sure the Decimal Separator is period (.) and the Thousand Separator is comma (,).
7. Click **Preview** to view the first few records of the data source.

   Each record in import_cube.csv contains 6 fields. The first five fields contain information that will be imported into TM1 as element names. The sixth column contains cube data.

Variable Name	Variable Type	Sample Value	Contents
V1	String	Actual	Ignore
Massachusetts	String	Argentina	Ignore
V3	String	S Series 1.8 L Sedan	Ignore
Units	String	Units	Ignore
Jan	String	Jan	Ignore
V6	Numeric	313.00	Ignore

Defining Cube Variables

After identifying the source data to TurboIntegrator, you must identify the contents of each field in the source.

Procedure

1. Click the **Variables** tab. TurboIntegrator sets default values for each variable.
2. For each variable, select a type from the associated Variable Type menu.

   In this example, no changes to the Variable Type fields are required. TM1 correctly identifies the type for each variable.
3. For each variable, select a content type from the associated Contents menu.

   In this example, all variables with the exception of V6 should be identified as Element. V6 should be identified as Data.

Mapping Cube Variables

You've identified variables for data, elements, and consolidations. Now you have to map the variables and provide instructions for creating a new cube.

Procedure

1. Click the **Maps** tab.
2. Click the **Cube** tab.
3. Select **Create** for the Cube Action.
4. Type **import_cube** in the Cube Name field.
5. Select **Store Values** for the Data Action.
6. Do not turn on the Enable Cube Logging option. When you enable cube logging, TM1 logs changes to cube data during processing. You are creating a new cube, so there is no need to log changes.

Mapping Cube Element Variables to Dimensions

Map all variables you identified as having an Element type to appropriate dimensions.

Procedure

1. Click the **Dimensions** tab.

2. Set values in the Dimensions tab according to the following table.

Element Variable	Sample value	Dimension	Order in Cube
Actual	Actual	actvsbud2	1
Argentina	Argentina	region2	2
V3	S Series 1.8 L Sedan	model2	3
Units	Units	measures	4
Jan	Jan	month2	5

3. For all element variables, set the Action to **Create**, and the Element Type to **Numeric** .

Mapping Cube Data Variables

For this example, there is only one data variable - V6. You do not need to map this data variable. TurboIntegrator does it for you. The data tab is not even enabled in this example.

TurboIntegrator adds the data to the cube at the intersection of the created dimensions. If there were 2 or more variables defined as data on the Variables tab, you would have to specify information about where the data should be added to the cube.

For an in-depth example of mapping data values into a cube, see "TurboIntegrator Tutorial."

Mapping Consolidation Variables

No variables in this example are defined as consolidations on the Variables tab. The Consolidations tab is not enabled in this example.

For an in-depth example of mapping consolidations into a cube, see "TurboIntegrator Tutorial."

Saving and Executing the Cube Process

You must save and name the process before you can execute it.

Procedure

1. Click the **Execute** button.

   To save and execute the process:

   TM1 prompts you to name and save the process.
2. Save the process as create_newcube.

   After a few seconds, you should see confirmation that the process executed successfully.
3. Open the Server Explorer. You should see that the cube import_cube has been created and populated, and that all required dimensions have been created.

Chapter 3. Importing from an ODBC Source

Using TurboIntegrator, you can create cubes and dimensions from data in relational database tables. To do this, you must have the following software on your machine:

- The client software for your relational database installed on the same machine on which you are running TurboIntegrator.
- An ODBC data source established for your relational database. You build data sources through the Windows Data Sources control panel.

Once you have defined the ODBC data source, the steps for creating a cube or dimension from relational data are identical to creating a cube or dimension from a text file. For a complete step-by-step tutorial of creating objects in TurboIntegrator using an ODBC source, see "TurboIntegrator Tutorial."

Note: TM1 requires DataDirect drivers to access an Oracle ODBC source on Solaris or AIX®. These drivers are not supplied with TM1 and must be acquired separately.

Unicode and DNS

When configuring the DSN to import Unicode data from an Oracle database using the version 11g client/ODBC driver, be sure to specify the Enable Closing Cursors option on the Application tab. TI processes can fail if this option is not specified.

The Oracle 11g ODBC driver does not adequately support the SQL_CLOSE option of the SqlFreeStmt.

Defining an ODBC Data Source

To define an ODBC data source:

Procedure

1. Open the Server Explorer.
2. Right-click the **Processes** icon beneath the server on which you want to create the process and choose **Create New Process**.

   The TurboIntegrator window opens.
3. Choose the top **ODBC** in the Data Source Type box. TurboIntegrator displays the fields required to define an ODBC source.
4. Click **Browse** and choose an ODBC data source name. Only data sources that have been defined on the computer on which the TM1 server runs are accessible.
5. If they are required to use this source, enter a valid username and password for the target database in the **UserName** and **Password** fields.
6. In the **Query** box, enter an SQL query to extract data from the source. The syntax and format of the SQL query depends on the type of database you are using. For example, if you are using an Microsoft Access Database, you can run Microsoft Access, open the database, use the SQL view, then copy the SQL statement into this Query window.

   Note: If the query references a table name that contains spaces, you must enclose the name in double quotes.
7. Click **Preview**.

   If the query was valid, and the connection was properly defined, the first ten records of the target database table appear in the TurboIntegrator window.

   See "Identifying Variables in the Data Source" for the steps used to define ODBC variables.

   See " Mapping Variables" for instructions on how to define ODBC map instructions.

See "Saving and Executing the TurboIntegrator Process" for details on saving and executing a TurboIntegrator process.

Generating a TurboIntegrator Process from an MDX Statement

This section describes how to extract data from an ODBO data source using an MDX statement, and import that data into TM1 .

It is best to generate an MDX statement using another utility, then use the working MDX statement as the basis for your data import into TM1 .

When you import data, it is important to start with an MDX statement that has a limited number of columns. Some MDX statements generate large numbers of columns. Such queries are impractical as a starting point for an import.

One method of limiting the number of columns is to place only the measures you are interested in on the columns.

Building the MDX TurboIntegrator Process

Once you have an MDX statement that returns useful data, you can build your TurboIntegrator process.

To get started, follow these steps:

Procedure

1. In the Server Explorer, right-click **Processes** and choose **Create New Process**. The TurboIntegrator window opens.
2. In the Data Source Type box, click **ODBO** and select **MDX Query**.
3. Enter required connection parameters in ther Connection tab of the TurboIntegrator window. The connection parameters are vendor-specific.
4. Click **Connect**. If you connect successfully, the Connect button becomes greyed out and you are able to proceed to the MDX Query tab.
5. Click the **MDX Query** tab.
6. Type your MDX query into this tab. You can also cut a working MDX query from another application and paste it into this tab.
7. Click the **Variables** tab. For each column generated by the MDX statement, one variable is generated by TurboIntegrator.

   Columns containing row headers are typically mapped as dimension elements. Columns containing data elements are mapped as data.
8. See "Mapping Variables" to map the variables into TM1 structures. Once you have connected to the ODBO data source and defined the MDX statement, the process for completing the TurboIntegrator process is identical to that of an ODBC data import.

Chapter 4. Importing from a TM1 View or Subset

IBM Cognos TM1 TurboIntegrator allows you to extract data from a cube view and create new objects with that data. The steps for building a process to use a TM1 view are similar to those used to define any other data source, except that you first build a view of your data that is specifically designed for import.

Not all TM1 cube views can be successfully imported. By building a view with certain parameters from inside TurboIntegrator, your import will work successfully every time.

Using a TM1 Cube View as a Data Source

You can define a cube view as a data source.

To do so, see "Create a Cube Process" on page 15 to define the data source, then follow the procedures described in "Importing a Text File".

You can also create a cube view using MDX expressions and use the MDX-based view as a TurboIntegrator data source. The following shows the column ordering to apply in a MDX-based view:

```
member1, member1_prop1, member1_prop2, ..., member1_propN, member2, member2_prop1, ...,
memberN_propN, cell_prop1, cell_prop2, ..., cell_propN, cell_value
```

Note: The definition uses the "_prop" notation to reference MDX dimension or cell properties, which are explicitly requested in the MDX query string.

The ViewCreateByMDX, ViewMDXSet, and ViewMDXGet TurboIntegrator functions are available to work with MDX views. For information on these functions, refer to the *TM1 Reference* documentation.

Create a Cube Process

You can create a process that uses a cube view as a data source.

Procedure

1. Right-click **Processes** in the Server Explorer and choose **Create New Process**.
2. Click **TM1** and select **Cube View** in the Data Source Type box. TurboIntegrator displays the Data Source Name field.
3. Click **Browse** to select from a list of available views. The Browse Server Cube Views dialog box opens.
4. Select the cube that holds the data you want to import.
5. If a view that you want to use as a data source already exists, select that view.

   If such a view does not exist, click **Create View** to open the View Extract window and create the view. After creating the view, select it in the Browse Server Cube Views dialog box.
6. Click **OK**.

   The selected view now displays as the data source for your TurboIntegrator process.

   Continue with the steps described in "Importing a Text File" to complete the import of your TM1 view.

Using the TM1 Subset as a Data Source

TurboIntegrator allows you to extract data from the TM1 dimension subset and move that information to another TM1 object. In the following example, the Europe consolidation in the Region dimension is extracted and used to form a new dimension called Region_Europe.

When you are extracting information from a dimension subset, the target object is typically another dimension. You cannot build a cube from information extracted from a dimension subset.

The procedure for extracting data using the TM1 Subset is similar to other TurboIntegrator process. See "Define Dimension Subset as Data Source" on page 16 to get started.

Define Dimension Subset as Data Source

Follow these steps to create a process that uses a dimension subset as a data source:

Procedure

1. Right-click **Processes** in the Server Explorer and choose **Create New Process**.
2. Click **TM1** and select **Dimension Subset** in the Data Source Type box. TurboIntegrator displays the single field required to define a cube view source.
3. Click **Browse** to select from a list of available subsets.

   The Browse Server Subsets dialog box opens.
4. Select the dimension that contains the elements you want to import.
5. Select the subset that you want to use as a data source and click **OK**.
6. Click **Preview**.

   The elements of the selected dimension subset appear in the preview panel.

Define Dimension Variables

In this example, the elements extracted from the subset data source will be added as children of a top-level consolidation called All Europe.

To build a new consolidation follow these steps:

Before you begin

See "Defining Cube Variables" for details on identifying and defining variables in TurboIntegrator.

Procedure

1. Click **New Variable**.

   The variable V2 displays in the Variables tab.
2. Click **Formula**.

   The Process Variable Formula dialog box opens.
3. Modify the formula as follows:

   ```
   V2='All Europe';
   ```
4. Click **OK**.
5. Change the Variable Type for V2 to **String**.
6. Change the Contents setting for V2 to **Consolidation**.

   In the next section, the elements imported from the subset data source are added to the All Europe consolidation.

Mapping Dimension Variables

In this example, you must set the Cube, Dimensions and Consolidations tabs to create a new dimension called Europe. Europe has a single consolidation called All Europe.

See "Mapping Variables" for details on the procedure for mapping imported data to TM1 objects.

Setting the Cube Tab

Set the following options on the Cube tab:

Action Type	Setting
Cube Action	No Action
Data Action	Store Values

Setting the Dimensions Tab

The Dimensions tab allows you to map incoming data into TM1 dimensions. In this example, only one dimension is created, named Europe. Set the following options on the Dimension tab:

Option Name	**Setting**
Element Variable	Europe
Dimension	Region
Action	Create
Element Type	Numeric

Setting the Consolidations Tab

The All Europe variable you added earlier should appear on the Consolidations tab. Notice that the Sample Value is set to the value you established in the formula. Because the process contains only two variables, TM1 correctly identifies the region variable as the child of the V2 variable. There is no need to modify the setting on the Consolidations tab.

Saving and Executing the Dimension

After saving and executing the process, TM1 creates a new dimension named Europe with a single consolidation named All Europe, which contains leaf elements for all European regions.

See "Saving and Executing the TurboIntegrator Process" for details on how to save and execute a TurboIntegrator process.

Chapter 5. Importing from MSAS

IBM Cognos TM1 TurboIntegrator allows you to import data from any OLE DB for OLAP (ODBO) data source, including Microsoft Analysis Services. This section shows how to use TurboIntegrator to import cubes and dimensions from Microsoft Analysis Services.

OLE DB for OLAP Data Sources

An OLE DB For OLAP data source is identified by the following parameters:

- ODBO Provider Name
- ODBO Location
- ODBO Data Source
- ODBO Catalog

ODBO Provider Name

This is the name assigned by the ODBO provider that identifies their multidimensional database server. For example, TM1 uses "TM1 OLE DB MD Provider" and Microsoft Analysis Services uses "Microsoft OLE DB Provider for OLAP Services 8.0".

TurboIntegrator lists only the ODBO providers that you have installed on your server.

ODBO Location

The location field is the name of the location where an administrator assigns a particular instance of the ODBO provider service.

The exact interpretation of this field is vendor-specific.

ODBO Datasource

This is 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.

ODBC Catalog

This is the name assigned by your administrator to a particular collection of databases (Cubes, Dimensions and other objects). For Microsoft Analysis Services, this is the name of the database.

Connection Strings: MSAS vs. TM1

The TM1 OLE DB for OLAP Provider has been modified to provide more flexibility to programmers building connection strings. This was done to make TM1 connection strings compatible with MSAS connection strings.

In earlier versions of TM1 , logging in through the TM1 OLE DB Provider required the following fields:

Field	Example Setting
Location The machine name of the TM1 Admin Server host.	MyServer
Datasource The name of the TM1 server.	Sdata
userID The TM1 user name.	Admin

Field	Example Setting
password The password for the TM1 user.	Apple

You can use the parameters described above, or you can log in to TM1 using the parameters in the following table. These parameters are also used to connect to Microsoft Analysis Services from TurboIntegrator.

Field	Example Setting
Datasource The machine name of the TM1 Admin Server host.	MyServer
Catalog The name of the TM1 server.	Sdata
userID The TM1 user name.	Admin
password The password for the TM1 user.	Apple

Connecting to an OLE DB for OLAP Data Source When Using CAM Authentication

If your TM1 server is configured to use Cognos Access Manager (CAM) authentication, you must specify the CAM namespace ID used by the server when establishing a connection to an ODBO data source.

You can specify the CAM namespace in the Additional Connection Parameters section of the Connection tab in TurboIntegrator. The CAM namespace ID must be specified using the following format:

```
Provider String="CAMNamespace=<CAM Namespace ID>"
```

<CAM namespace ID> must be the internal CAM namespace ID, not the descriptive name of the namespace.

For example, the following table shows a CAM namespace ID, NTLM_NAMESPACE, included in the value of the Additional Connection Parameters property.

Property	Value
ODBO Provider	IBM Cognos TM1 OLE DB MD Provider
ODBO Location	localhost
ODBO Datasource	Planning Sample
ODBO Catalog	
ODBO UserID	*user@mycompany.com* (the CAM user name on the source TM1 server)
ODBO Password	*password_for_CAM_UserID*

Property	Value
Additional Connection Parameters	Provider String="CAMNamespace=NTLM_NAMESPACE";InitialCatalog=empty

Importing a MAS Cube

This procedure describes how to import a simple cube from Microsoft Analysis Services into TM1 .

To import a cube into TM1 from Microsoft Analysis Services:

1. Establish the connection to the MAS data source.
See "Connecting to Analysis Services with TurboIntegrator" on page 21.
2. Specify which cube you are importing.
See "Specifying the Cube with the Load ODBC Cube Tab" on page 22.
3. Define the dimensions.
See "Using the Cube Dimensions Tab" on page 22.
4. Save the process and run it.
See "Saving and Executing the MAS Process" on page 23.

Connecting to Analysis Services with TurboIntegrator

Use TurboIntegrator to create a process that connects to Microsoft Analysis Services.

Procedure

1. Run Architect, and log in using a valid user name and password.
2. Right-click **Processes** and choose **Create New Process**.

   The TurboIntegrator dialog box opens.
3. Click the **ODBO** option and then select **Cube**.

   The dialog box displays the options that allow you create to an ODBO connection string.
4. Enter connection parameters into the dialog box as follows:

Field	Value
ODBO Provider	Choose **Microsoft OLE DB Provider for OLAP Services**.
ODBO Location	Leave this parameter blank.
ODBO Datasource	Enter the machine name of the server that hosts Analysis Services.
ODBO Catalog	Enter an Analysis Services database name. For example, to import data from the Microsoft sample database, enter **FoodMart 2000** in this field.
ODBO UserID	Enter a valid user name for the Analysis Services database.
ODBO Password	Enter a valid password for this username for the Analysis Services database.
Additional Connection Parameters	Some ODBO servers may require additional parameters in order to successfully connect. Enter those parameters in this field, delimited by semi-colons.

5. Click **Connect**. If you connect successfully, the Connect button becomes greyed out, and you are able to proceed to the Load ODBO Cube tab.

Specifying the Cube with the Load ODBC Cube Tab

The Load ODBO Cube tab allows you to specify which cube you are importing from Analysis Services, along with other information. Follow these steps to fill out this tab.

Procedure

1. Click the **Load ODBO Cube** tab.
2. Choose a cube action. The choices are described in the following table:

Option	Description
Create Cube	Copies data and metadata from the ODBO data source, and create a new cube in TM1 . Use this option only when none of the cubes and dimensions you are importing exist on the server.
Recreate Cube	Destroys a currently existing cube, and rebuilds it using data and metadata from the ODBO data source. Use this option only when the cubes and dimensions exist, and you want to replace them with new structures and data.
Update Cube	Copies data from an existing ODBO cube, and inserts it into an existing cube. This option does not change the structure of cubes and dimensions on the server.
No Action	The default value for the screen. Processes that specify No Action do not affect the cube's data or metadata. Use this to test and debug processes or to define your own custom operations.

   For this example, choose **Create Cube**.
3. Click **Select ODBO Cube From** and choose an Analysis Services cube to import into TM1 .
4. Click in the **Select TM1 Cube to Load To** field. Enter a unique name for your cube.
5. In the Data Action panel, choose **Store Values**. This option writes cell values in the ODBO cube to the cube. The Accumulate Values option allows you to aggregate values as they are being imported.

Using the Cube Dimensions Tab

The Cube Dimensions tab allows you to manipulate imported dimensions as they are imported into TM1 .

By default, all of the dimensions in the ODBO cube are imported. They are created in TM1 as *name_*. For example, when the [customer] dimension in Analysis Services is imported, the corresponding dimension in TM1 is called Customer_.

This dialog box presents the following options:

- You can choose to map an ODBO dimension to an existing dimension. To do this, click any dimension in the **TM1 Dimension** column, and choose another dimension.
- You can also import the ODBO dimension's elements into an entirely new dimension. Click in the corresponding cell underneath the TM1 Dimension column, then type the name of the new dimension. For example, replace the customer_ dimension with a dimension called MyCustomerDim.
- For each imported dimension, you must choose the TM1 Dimension Action. Choose one of the following options:

Option	Description
Create	Imports dimension data from the ODBO cube, and creates a new dimension with the entire set of element from the dimension. This is the default action.
Filter Only - MDX	Imports dimension data from the ODBO cube, and creates a new dimension in with a limited set of elements.

Option	Description
No Action	Do not import this dimension from the ODBO data source.

Saving and Executing the MAS Process

Once you have completed your changes to the Cube Dimensions tab, click ⚡ to save and execute the process.

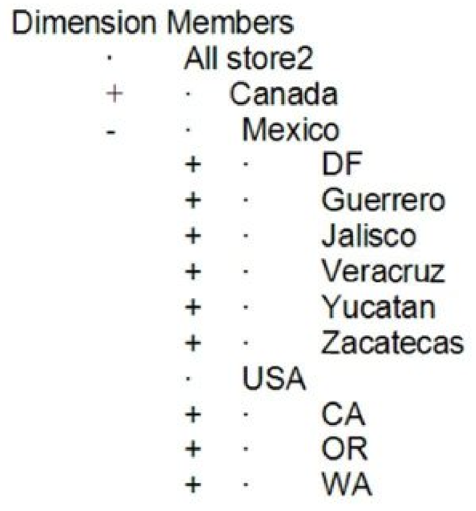
The Save Process As dialog box opens.

Enter the name of the new process. Give the process a name related to the data you are importing. For this example, enter **ODBO_Sales_Import**.

TM1 should import your data and create the new cube. A dialog box will appear to show the progress of the import.

Importing a MAS Dimension

This section describes how to import a dimension from Microsoft Analysis Services into TM1 . The following table is a representation of the dimension as displayed in Analysis Services.



```
Dimension Members
        ·       All store2
        +   ·     Canada
        -   ·      Mexico
                +   ·     DF
                +   ·     Guerrero
                +   ·     Jalisco
                +   ·     Veracruz
                +   ·     Yucatan
                +   ·     Zacatecas
                    ·  USA
                +   ·     CA
                +   ·     OR
                +   ·     WA
```

Figure 1: Sample dimension

TM1 requires that all elements in a dimension have unique names. TM1 also requires that all aliases for the elements have unique names. In order to ensure that the element names are unique, TM1 names each consolidation and element in an imported dimension with the names of all its parents in square brackets, delimited by periods.

After being imported into TM1 , the subset aliases are populated with the element names from Analysis Services.

The procedure to import MAS data is similar to other import processes.

Define MAS Connection Parameters

The first step in importing an Analysis Services dimension into TM1 is connecting to Analysis Services and choosing the ODBO Dimension option. Follow these steps:

Procedure

1. Run Architect, and log in using a valid user name and password.
2. Right-click **Processes** and choose **Create New Process**.

   The TurboIntegrator dialog box opens.
3. Click the **ODBO** option and then select **Dimension**.

4. Enter connection parameters into the dialog box as follows:

Field	Value
ODBO Provider	Choose **Microsoft OLE DB Provider for OLAP Services**.
ODBO Location	Leave this parameter blank.
ODBO Datasource	Enter the machine name of the server that hosts Analysis Services.
ODBO Catalog	Enter an Analysis Services database name. For example, to import data from the Microsoft sample database, enter **FoodMart 2000**.
ODBO UserID	Enter a valid user name for the Analysis Services database.
ODBO Password	Enter a valid password for this user for the Analysis Services database.
Additional Connection Parameters	Leave this field blank.

5. Click **Connect**. The Connect button should grey out, indicating that you connected successfully.

Using the Load ODBO Dimension Tab

Once you are successfully connected to Analysis Services, you must specify information about the source and destination dimensions for your dimension load process. Follow these steps:

Procedure

1. Click the **Load ODBO Dimension** tab.
2. Choose the TM1 Dimension Action. Choose one of the following options:

Option	Description
Create Dimension	Copies a dimension from the ODBO data source, and creates a new dimension.
Recreate Dimension	Destroys a currently existing dimension, and rebuilds it using data from the ODBO data source.
Update Dimension	Update Dimension assumes that TM1 already has a dimension into which you want to insert or delete elements. • If elements exist in the ODBO data source but not in TM1 . The elements are added to the dimension. • If elements exist in TM1 , but not in the ODBO data source, those elements are unaffected by the import. No changes are made to the elements in the local dimension. • If elements exist in the ODBO data source and the local dimension, the elements from the ODBO data source are imported, and they are created in the local dimension as <element_name>_1. Note that this will increase the size of your dimension.
No Action	The default value for the screen. This process has no effect on the dimension.

3. Click the **ODBO Cube Containing Dimension** list and choose the cube that contains the dimension that you want to import from Analysis Services.
4. Click the **Cube Dimensions** list and choose the dimension you want to import.
5. If you are updating or recreating a dimension, click the **TM1 Dimension to Load** list and select a dimension from the list.

If you are creating a new dimension, type in the name of your new dimension in the TM1 Dimension to Load field.

Save and Run the Dimension MAS Process

Once you have completed your changes to the Load ODBO Dimension tab, click the Execute icon to save and execute the process.

The Save Process As dialog box opens.

Enter the name of the new process, then click **Save**. The import begins, and TM1 displays a dialog box showing the status of the import.

TM1 Message Log

When the process completes, minor errors may be written to the TM1 message log. If so, TM1 displays a message box to inform you.

To check the server message log, right-click the TM1 Server in the Server Explorer, and choose **View Message Log**. To see details about an error, double-click the error in the message log.

Chapter 6. Importing Data Using the IBM Cognos TM1 Package Connector

The IBM Cognos TM1 Package Connector is currently supported for use with IBM Cognos Business Intelligence packages against relational and ODBC data sources.

See the Supported Hardware and Software link on the (http://www.ibm.com/support/docview.wss?uid=swg27040698) for specifics on supported software.

When importing packages using the TM1 Package Connector. do the following tasks:

- Create a Package in Framework Manager.
- Create a TurboIntegrator process that uses the TM1 Package Connector.

This topic describes using the TM1 Package Connector.

The IBM Cognos TM1 Package Connector does not work with the 64-bit versions of TM1 Perspectives or TM1 Architect.

The IBM Cognos Package Connector is an optional component stored on a CD separate from the main TM1 installation disk. See "Installing the IBM Cognos TM1 Package Connector" in the *Planning Analytics Installation and Configuration* documentation for details on installing and configuring the IBM Cognos TM1 Package Connector.

Establishing a Connection to the Cognos BI Server

After the components have been installed and configured, follow these instructions to establish the connection:

Procedure

1. Run TM1 .
2. In the left pane of the TM1 Server Explorer, right-click the **Processes** group and select **Create New Process**.

   The TurboIntegrator window opens.
3. Select the **IBM Cognos Package** option in the Data Source Type dialog box.
4. **Package** is selected by default.

   Remember:

   Package and Dimension

   The Package and Dimension option provides a simplified way to import data from dimensionally modeled sources and is not easily customized.

   You can instead select **Dimension** if you only want to import hierarchies from a single dimension in a package.

   Custom Query

   The Custom Query option works with DMR or non-DMR sources and provides more flexibility in selecting the items to query and provides more open access to the standard TurboIntegrator programming capabilities.

   See Connecting to Published Packages for more details.
5. Enter the connection details:

   If your IBM Cognos BI server permits anonymous logon, you can click "Log on." Otherwise, enter your logon credentials here:

   - **Authentication Namespace**

     The pull-down list shows all available authentication namespaces.
   - **User ID** and **Password**

     Supply a user ID and password for a user in the selected authentication namespace.
   - **Sign-ons**

Allows you to manage sign-on information associated with the TurboIntegrator process. The button is enabled once you have successfully logged into the BI Server.

A sign-on called "BI Server Logon" is automatically created for you the first time you authenticate to the BI Server. If you update the credentials in the Connection tab after the initial login, those changes do not update this sign-on. You must use the Manage Sign-ons dialog box to make changes to the sign-on. You can Add, Modify, and Remove sign-ons. Also, if you enter credentials on the Connection tab, then click on logon and modify the sign-on in the sign-on window, the credentials on the Connection tab are not updated.

Removing a sign-on can make the TurboIntegrator process unable to run when anonymous logon is turned off.

As you work with dimensions and measures, data source sign-ons may be created. These can also be managed through the Sign-ons window.

- **Log on/Log off**

  Click **Log on** to log in to the BI server. Once you are logged in, Log off becomes available to let you log out of the BI server.

Connecting to Published Packages - Package and Dimension option

Once you have connected to the BI server, move to the Package tab and select the package that you want to work with.

If you are not importing measures, after selecting a Package, proceed to the Dimension tab.

Procedure

1. Click the **Browse** button to select from available packages. Only packages containing dimensions are available for selection.

   When you select a package from the Browse Metadata window, the **Select Package** field is automatically filled in for you. The Edit button becomes available after a package is selected.
2. Select the **TM1 cube to load to** option. If you are creating a cube, enter the cube name. If you are recreating or updating an existing cube, select it from the list.
3. Specify the import actions to take.

   Complete the Cube Action and Data Actions options as needed for this import, just as you would for any other type of datasource. See Mapping Variables for details on these options.
4. To define the dimensions in the package, click the **Dimension** tab.
5. Define the Hierarchy and Attribute structure using the dialog boxes offered when you click **Select Hierarchies** or **Select Attributes**.

   Keep in mind the following details about Hierarchies and Attributes:

   - **Default Hierarchy**

     If you specify the TM1 Dimension without pre-selecting any hierarchies, TM1 will automatically select the first one as the Default Hierarchy. You can change the selection by unchecking or checking the check boxes.
   - **Selecting before mapping**

     You can also select hierarchies and all sub items such as filters *before* mapping to a dimension. When you fill in the dimension on the pre-selected hierarchies dimension, all the selections are applied automatically. If you unselect any mapped dimension by clearing the dimension name in the field, all the pre-selected hierarchies under the corresponding dimensions are automatically cleared.
   - **Name versus Reference**

     Hierarchy displays Hierarchy Name and Hierarchy Reference to clearly identify the specified hierarchy. Hierarchy name can be repeated but hierarchy reference is unique.
   - **Filters**

     If a filter is defined in the package, and a hierarchy is selected, the Select Filters button becomes available so you can choose filters to apply.

     Ensure the filter makes sense for the selected hierarchy or your query may fail to execute.
   - **Multiple Levels in a Hierarchy**

IBM Cognos allows users to define multiple levels in a hierarchy. When a hierarchy defines multiple levels, attribute names may be repeated across different levels.

For example, in a dimension called City, it could contain City, Geography, Geography_link, etc. Each hierarchy can define multiple levels.

You can have, for example, dimensions such as Level Label, Level Number, Hierarchy Unique Name, etc. that are the same across different levels under different hierarchies. TM1 performs the consolidation of the attributes based on the two factors: attribute external name and attribute roles.

The precedence order is attribute external name and then attribute roles. If any attribute contains the same Attribute External Name values, those attributes are consolidated.

If an attribute does not contain the external name, its roles signature (where all the roles compound to a role signature) will be evaluated as the factor of the consolidation. In this context, "compound" refers to items with identical roles being grouped together in a single TM1 attribute

When specifying the attribute mapping, all the attributes whose external name or role signature matches with the mapped attributes are selected during importing. More than one attribute may be mapped.

6. **Select Attributes**

   Map a dimension attribute to a TM1 attribute by doing one of the following:

   - To map an attribute to a new attribute, enter a name for the new attribute in the corresponding TM1 Attribute field, then select an Attribute Type. The new attribute will be created when you execute the TurboIntegrator process.
   - To map an attribute to an existing attribute, click the corresponding **TM1 Attribute** field, select an attribute, then select an **Attribute Type**.

Attribute Type	**Description**
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. You can use this attribute to display dimensions by their alternative names in the TM1 user interface. A dimension alias must be unique from all other dimension aliases or actual dimension names.

   You must now map the measures to a dimension.

   The measures appear as the last row in the Dimensions tab.

   Measures have to be selected by clicking on Select Measures. TurboIntegrator does not pre-select the measures as there are no default ones.

7. Click **Select Measures**.

   The Select Measures dialog box opens.

   Note: A package may have multiple measure dimensions. The name of the measure includes the measure dimension name from the package.

8. Select each measure you want to import into TM1 .
9. Click **Filters** to select filters to apply to the measure query. Ensure the filter makes sense for the selected hierarchy or your query may fail to execute.
10. Click **OK**.
11. Map the measures to a dimension.

    If the measures map to an existing dimension, click the **TM1 Dimension** column and select the dimension that corresponds to the measure.

    If you want to create a new dimension from the measure, enter a name for the dimension in the TM1 Dimension column.

12. Select a TM1 Dimension Action for the measures.
13. **Dimension Settings**

    Use the Dimension settings tab to define the Top Consolidation for any specified dimensions.
14. **Prompt Editing**

    You can click the Prompts button to open a Cognos Prompt editing window to set and change the prompt values. If mandatory prompts exist in the package, prompt values must be provided before the query is run by TurboIntegrator process. Otherwise the process will fail.

    You can use the Prompts button to use the UserInterface to allow setting values or ranges. Click in the cell under the **Value** column to set the prompt value.

    Or you can use the following TurboIntegrator processes/APIs.

    If you know the prompt names, you can directly call CGAddPromptValues by passing the prompt name and values. These functions must be entered in the Prolog of the Parent TurboIntegrator process.

    These functions allow you to retrieve prompts that are defined in the TurboIntegrator process. To define them in the underlying TurboIntegrator process, you must have launched the Prompts window in your TurboIntegrator process. Click **Prompts**, pull the prompts from all defined queries, and click **OK** to make them available. Anytime you make changes to the prompts you must click **OK** to make the prompts available.

TurboIntegrator process API	**Description**
CGPromptSize()	returns the total number of prompts that need to be set
CGPromptGetNextMember(int index)	returns the prompt by index (0-(CGPromptSize()-1))
CGAddPromptValues(promptName, value 1, value 2...)	sets the prompt values by specified prompt name

Example	`count=CGPromptSize();`
	`while(i<count);`
	`  prmptname=CGPromptGetNextMember(i);`
	`  CGAddPromptValues(prmptname,`
	`    '1999-01-01','2009-01-01');`
	`  i=i+1;`
	`end;`

15. When there are mapped dimensions, hierarchies and measures, you can test the query by clicking **Test Queries**.

    Test Queries triggers a query execution before the TurboIntegrator process run. It can be used to pre-test if the queries run successfully and to supply missing prompt values or sign-on information.
16. Click the **Show Namespace** checkbox to include the namespace in the Dimension listing.
17. Complete the Advanced and Schedule tabs if needed. See Editing Advanced Procedures and Chores for details. The Package Connector also generates the TurboIntegrator process for each dimension. This gives you the flexibility to add your own TurboIntegrator process scripting statements if needed.
18. Save and execute the TurboIntegrator process. See Saving and Executing the TurboIntegrator Process.

Multiple hierarchies

The IBM Cognos TM1 Package Connector includes only one hierarchy per dimension when issuing a measure query.

If you select more than one hierarchy for a dimension and include measures in your TurboIntegrator process, the TM1 Package Connector will issue multiple measure queries, substituting each hierarchy in turn. This behavior is appropriate where the leaf-level members of each hierarchy are distinct from each other.

If the leaf level members are the same in the hierarchies, you should create a TurboIntegrator process selecting only one hierarchy per dimension along with the desired measures. Create a separate process to merge multiple hierarchy structures into the same dimension.

Importing a Single Dimension

You can use the Dimension pull-down option on the Data Source tab as a quick way to define a single dimension.

Choose **Dimension** from the IBM Cognos Package option pull-down to directly open the Dimension tab and specify the actions to take and to define the hierarchies.

Procedure

1. Locate the package.

   Click the **Browse** button to select from available packages.
2. Identify the **Dimension to load from:**.

   The pull-down lists the dimensions available in the selected package.
3. Identify the **TM1 Dimension to load into:**

   The pull-down lists the dimensions available if you are updating an existing dimension.
4. Select a **TM1 Dimension Action** just as you would for any other datasource. See Mapping Dimensions for details.
5. Identify the **Top Consolidation**:

   If you want to create a top-level consolidation for the dimension, enter the name of the top-level consolidation here.

   The resulting dimension will include a consolidation with the name you entered. For example, if you enter *Total* in **Top Consolidation**, the dimension includes a top level consolidation named Total with all imported elements as children of the consolidation.
6. When there are mapped dimensions, hierarchies and measures, you can test the query by clicking **Test Queries**.

   Test Queries triggers a query execution before the TurboIntegrator process run. It can be used to pre-test if the queries run successfully and to supply missing prompt values or sign-on information.

Connecting to Published Packages - Custom Query option

You can use the Custom Query option of the IBM Cognos TM1 Package Connector to connect to any kind of dimensionally modeled source to create a customized query. The Custom Query is the only way to connect to a non-dimensionally modeled source.

About this task

After connecting to the IBM Cognos Business Intelligence (BI) server, specify the source package you want to work with and the fields in that source package that you want to import to IBM Cognos TM1 in this process.

Procedure

1. Click the **Browse** button to select from available packages.

   When you select a package from the Browse Metadata window, the **Select Package** field is automatically filled in. The **Edit** and **Edit Query** buttons become available after a package is selected.

   The **Edit** button shows information about the data source and allows you to create a custom sign-on if the content store sign-on requires a password.
2. Click on the **Edit Query** button. The fields available in the source package are shown.
3. Select the each field required for extraction and select **Add**.
4. Choose the **Query Options**:

   Preview
      Shows a columnar lists. MDX displays the code used for the transformation.

   Match
      links a query item from the source package or report to a column in the query. It is only needed if the source has been modified and a query item has been moved or renamed. The Match option allows you to match the columns and query items back up again after moving or renaming an item.

Auto Summarize
: Selected by default. The query generates SQL to aggregate duplicate rows. For queries based on relational packages, enabling the Auto Summarize feature also helps reduce the number of rows that Cognos TM1 Package Connector retrieves from the source data, further improving cube build performance. If the source data is being used at the same granularity as the underlying table this should remain unchecked. If the source data is not consolidated then this should be checked. Ensure that the query has appropriate identifier and fact usage attributes set for this setting to be effective. These settings need to be set in the source; either Cognos Framework Manager or the report. Review the SQL to ensure appropriate grouping and summary functions are being applied. Do not use Auto Summarize to compute an average, but normally using Auto Summarize is desirable.

Supress Null values
: By default this option is set to use the governor setting in the defined package. To override this setting, select either **Yes** (Suppress Null values) or **No** (do not suppress).

5. Select the **Validate** button to verify that a valid query can be generated from the information selected.
6. Click **OK** to enable the **Prompts** and **Preview** button.

   Prompts
   : Click **Prompts** if any prompt values need to be set.

   Preview
   : Click **Preview** to see the MDX that will be generated and to preview the data if desired.
7. Click on the **variables** tab. From this point forward, the procedure is the same as for any other TurboIntegrator process. See "Identifying Variables in the Data Source" on page 6.

Saving, Executing, and Editing the TurboIntegrator Process with the TM1 Package Connector as a datasource

To save, edit and execute the process.

About this task

When you create a TurboIntegrator process using the TM1 Package Connector as a datasource, TurboIntegrator stores the encrypted password for the Business Intelligence authentication namespace.

When you edit a TurboIntegrator process that uses the TM1 Package Connector as a datasource, you must re-enter the credentials required to access the Business Intelligence Server.

Procedure

Use these steps to create a TurboIntegrator process.

1. Click the **Execute** button.

   TM1 prompts you to name and save the process.
2. Save the process.

   You should then see confirmation that the process executed successfully.
3. Open the Server Explorer. You should see that the cube you specified has been created and populated, and that all required dimensions have been created.

   - If you are editing a TurboIntegrator process, you must first re-enter IBM Cognos BI credentials required to access the Business Intelligence Server.
   - If you are creating a process, open the tm1s.cfg and add the following configuration parameter: EnableTIDatasourcePasswordAccess=T.

Seeing the Results

When you define a process to import a package into TM1 and create a new cube, the following actions occur:

- For each dimension you choose to include in your cube, TM1 generates a process to import the dimension and create a corresponding dimension.
- TM1 generates a master process that executes the above described dimension-creating processes, builds the cube, and imports data values.
- The query item defined as the key for the level in the hierarchy is used as the element name.

Chapter 7. Editing Advanced Procedures

This section describes managing IBM Cognos TM1 TurboIntegrator processes.

Using Bulk Load Mode

Bulk Load mode enables TM1 to run in a special optimized single-user or single chore/process mode. This mode can maximize performance for dedicated tasks during times when little or no other activity is expected.

Some examples of using Bulk Load mode include:

- An administrator that needs to manually perform maintenance operations.
- A night-time window to load large amounts of data.

TM1 typically runs in a multi-user mode where multiple users, chores and processes can all run concurrently accessing data. In Bulk Load mode, the TM1 server prevents concurrent activity by temporarily suspending other users, chores and processes and eliminates the overhead required by a multi-user environment.

Bulk Load mode doesn't actually log out users, but only suspends their interaction with TM1 . As soon as Bulk Load mode is done, any users that were previously logged in are reactivated and user-interaction with TM1 resumes.

You can enable Bulk Load mode directly within a TI process or with the TM1 API. In either case, you use commands to *enter* and *leave* Bulk Load mode.

Considerations for Using Bulk Load Mode

You should consider the following when using Bulk Load mode:

- Bulk Load mode does not display a message to end-users to alert them. You will need to plan and coordinate your usage of Bulk Load mode accordingly.
- Only a single user or process may be active during Bulk Load mode. No new connections can be established to the server while it is operating in Bulk Load mode.
- A TI process can not use the `ExecuteCommand` to launch a command line program that attempts to log back into the same TM1 server. The login attempt will fail.
- Any scheduled chores that are scheduled to execute during the time Bulk Load mode is enabled are deactivated and not run.

Starting Bulk Load Mode

When the server enters Bulk Load mode, all processing by other threads is paused. Any existing user threads and running chores will be suspended. Only the thread that initiated Bulk Load mode will remain active. All scheduled chores will be deactivated, except the chore that initiates Bulk Load mode. All system-specific threads and TM1 Top connections will also be suspended.

Ending Bulk Load Mode

When Bulk Load mode is disabled, all system and user threads will be resumed and user logins will be allowed.

Custom applications that use the TM1 API to enable Bulk Load mode should also call the necessary TM1 API function to *exit* Bulk Load mode. However, if the client connection is severed (the network fails or the client logs out, crashes or disconnects), the server will automatically exit Bulk Load mode.

Similarly, if a TI process/chore is running in Bulk Load mode and the process exits, whether successfully or with errors, the server will automatically exit Bulk Load mode.

When the server returns to normal multi-user mode, any chores that were deactivated get reactivated and return to their normal schedule. If the chores were scheduled to run, but were prevented by Bulk Load mode, they will not get executed immediately, but will execute according to the schedule. You may have to adjust the launch time of your scheduled chores to prevent them from getting locked out during the times you enable Bulk Load mode.

TurboIntegrator Process Commands for Bulk Load Mode

You can enable Bulk Load mode in either the Prolog or Epilog section of a TI process. For efficiency, we recommend enabling 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 TI 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 TI 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 TI commands to enable and disable Bulk Load mode in a TI 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.

TM1 C API Functions for Bulk Load Mode

The following TM1 C API functions are available for enabling and disabling Bulk Load mode.

- `TM1ServerEnableBulkLoadMode`
- `TM1ServerDisableBulkLoadMode`

For details, see the IBM Cognos TM1 *API* documentation.

Editing Procedures

After you specify a data source, identify all variables, and define all mapping instructions, TurboIntegrator generates four procedures that are based on the options you selected in the TurboIntegrator tabs. These procedures are identified as sub-tabs of the Advanced tab.

The procedures are:

Tab	**Description**
Prolog	A series of statements to be executed before the data source is processed.
Metadata	A series of statements that update or create cube, dimensions, and other metadata structures during processing.
Data	A series of statements that manipulate values for each record in the data source.
Epilog	A series of statements to be executed after the data source is processed.

You can edit these procedures to include TurboIntegrator functions and TM1 rules functions that extend the capabilities of TurboIntegrator. For example, you can edit the Data procedure to include statements that instruct the process to skip records containing zero values, or to write imported records to an external file.

For a complete list of all available TurboIntegrator and TM1 rules functions, see the IBM Cognos TM1 *Reference*.

When editing procedures, keep in mind that each procedure is intended to execute certain types of actions at specific times in a process. Accordingly, you should create actions or statements that are appropriate for a given procedure.

Note: When the data source for a process is NONE, the Data and Metadata procedures are ignored when the process is executed. Any functions or statements on the Data or Metadata sub-tabs are not executed, but TM1 does not issue an error or warn you that part of the process was not executed.

To edit a procedure:

Procedure

1. Click the **Advanced** tab.
2. Click the sub-tab for the procedure you want to edit.
3. Enter your statements in the text box either *before* this line:

   ```
   #****GENERATED STATEMENTS START****
   ```

   or *after* this line:

   ```
   #****GENERATED STATEMENTS FINISH****
   ```

   Important: User-created statements can be inserted either before or after the generated statements, but cannot be inserted within the statements generated by TurboIntegrator.

Executing a Process on Demand

To execute a process on demand, select the process in the Server Explorer and choose **Process**, **Execute Process**.

You can also execute a process from within TurboIntegrator by choosing **File**, **Execute** ⚡.

Using TM1RunTI

TM1RunTI is a command line interface tool that can initiate an IBM Cognos TM1 TurboIntegrator (TI) process or chore from within any application capable of issuing operating system commands.

This utility is of special interest in application situations where TurboIntegrator processes and chores need to be grouped in order to ensure that the processes/chores run in parallel. It is also helpful so that those processes and chores which cannot be run in parallel are serialized in the right order. Note that TM1RunTI does not finish (return) before the TurboIntegrator is finished which can be used to serialize calls if the calling process or chore is waiting for TM1Runti to finish.

The TM1RunTI executable file (tm1runti.exe) can be found in the bin directory of a TM1 server install.

If you accepted the default installation location for tm1runti.exe is in `C:\Program Files\IBM\cognos\tm1\bin`.

Asynchronous calls and TM1

The Execute command takes two parameters; the second one describes whether to have a synchronous call or an asynchronous call. Cognos TM1 tools should only be called asynchronously (Parameter 0) to avoid server deadlocks if the system is waiting for a lock held by the TurboIntegrator process/chore and the process/chore is waiting for the utility. The same advice applies to any executables called by ExecuteCommand if they login to Cognos TM1 .

Note: Never use a synchronous call if the tool logs into Cognos TM1 .

TM1RunTI syntax

The TM1RunTI syntax is described here.

```
      tm1runti -?
                 or tm1runti -help
                 or tm1runt1 [<cmd_parm>...] [<ti_parm>...]

                    where <cmd_parm> is one of:
                         -i <filespec>
                         -process <string>
                         -chore <string>
                         -connect <string>
                         <connect_parm>...

                    where <ti_parm> is:
```

```
                         <parm_name> '=' <parm_value>

                    where <connect_parm> is one of:
                         -adminhost <string>
                         -server <string>
                         -user <string>
                         -securitymode
                         -retryattempts
                         -retryinterval
                         <password_parm>
                         -AdminSvrSSLCertAuthority <filespec>
                         -AdminSvrSSLCertID <id>
                         -AdminSvrSSLCertRevList <filespec>
                         -AdminSvrSSLExportKeyId <id>
                         -ExportAdminSvrSSLCert <T|F>
                         -CAMNamespace <string>

                    where <password_parm> is one of:
                         -pwd <string>
                         -passwordfile <filespec> -passwordkeyfile <filespec>
```

Parameters

Parameters can be either in a configuration file or passed on the command line. Command line parameters take precedence over parameters that are in the configuration file. This makes it possible to have persistent default parameters for relatively static parameters (such as adminhost and server) and to supply just the few parameters needed to either override the defaults or to provide values that are not easily defaulted, such as the user name or the TurboIntegrator process name.

The parameters have a different format when passed on the command line. While all parameters are passed in a "-parameter_name value" fashion, everything that is passed as "parameter_name=value" is treated as a TurboIntegrator process parameter.

There are four types of parameters:

- Command parameters

  Used to specify the config file to use, which group of connection parameters to use, which TurboIntegrator process to run, or which TurboIntegrator chore to run.

  Only one command parameter should be specified at a time. If both a -process and -chore are present on the command line, TM1RunTI will run the TurboIntegrator process and ignore the -chore parameter.
- Connection parameters

  Used to specify the servername, username and other information needed to connect to the Cognos TM1 server.
- Password parameters

  Can either be a username and plain text password or can be a filename containing an encrypted password and associated keyfile used for decryption.
- TurboIntegrator parameters

  Passed to the named TurboIntegrator.

Parameters specified on the command line must begin with dash (-) or slash (/). The parameter value is separated from the parameter name by a space, and the value can be specified as is or in quotes (if there are embedded spaces).

For example:

```
tm1runti –server MyTM1Server –username John –pwd "my secret"
         ti_parm1=yes ti_parm2="my value"
```

TM1RunTI Parameters

Parameter	Description Value/Required/Default
`i`	Path to configuration files String/No/None
`connect`	This parameter can be used to specify a section in the configuration file containing parameters used to make server connections, such as user, pwd, CAMnamespace, etc. String/No/None
`process`	Name of the TurboIntegrator process to call. String/No/None
`chore`	Name of the TurboIntegrator chore to call. String/No/None
`help`	Display help text to the command window (stdout). not applicable/No/not application
`?`	Display a synopsis of command line parameters to the command window (stdout). not applicable/No/not application

Connect Parameters

Connect parameters are common among TM1 tools, and can be defined in their own section to enhance reuse and avoid the effort and risks associated with maintaining multiple copies.

Parameter	Value/Required/Default	Description
`adminhost`	String/No/None	Cognos TM1 admin host
`sever`	String/No/None	Cognos TM1 server name
`user`	String/No/None	Cognos TM1 or CAM name
`retryattempts`	Numeric/No/none	Number of attempts to connect to the TM1 server when the server is unavailable due to temporary network issues or instances when the TM1 Server is not able to handle incoming connections. If the TM1 server is not running, any connection attempt fails and returns an error. There are no further connection attempts, regardless of `retryattempts` parameter setting.

Parameter	Value/Required/Default	Description
`retryinterval`	Numeric/No/0	Time interval, in seconds, to retry connection to the TM1 server. The number of connections attempted is determined by the `retryattempts` parameter value. When `retryinterval` is not specified or is set to 0 (default), connection attempts occur immediately without a wait interval.
`AdminSvrSSLCertAuthority`	String/No/none	The full path of the certificate authority file that issued the Cognos TM1 Admin Server's certificate
`AdminSvrSSLCertID`	String/No/none: API Default is : `tm1adminserver`	The name of the principal to whom the Cognos TM1 Admin Server's certificate is issued. **Note:** The value of this parameter should be identical to the SSLCertificateID parameter in the `Tm1admsrv.ini` file.
`AdminSvrSSLCertRevList`	String/No/None	The full path of the certificate revoca-tion file issued by the certificate authority that originally issued the Cognos TM1 Admin Server's certificate. A certificate revocation file will only exist in the event a certificate had been revoked.
`ExportAdminSvrSSLCert`	Boolean/No/F	Specifies whether you want the certificate authority certificate which originally issued the Cognos TM1 Admin Server's certificate to be exported from the Microsoft Windows certificate store at runtime. When this option is selected, you must also set a value for `AdminSvrSSLEx-portKeyID` as described here. Refer to *IBM Planning Analytics Installation and Configuration* for appropriate TM1Server configuration.
`AdminSvrSSLExportKeyId`	String/No/None	The identity key used to export the certificate authority certificate, which originally issued the Cognos TM1 Admin Server's certificate, from the certificate store. This parameter is required only if you choose to use the certificate store by setting `ExportAdminSvrSSLCert=T`. Refer to *IBM Planning Analytics Installation and Configuration* for appropriate TM1Server configuration.
`CAMNamespace`	String/No/none	CAM namespace id. **Note:** This not the CAM namespace name. This value is needed only if the Cognos TM1 Server authenticates using CAM.

TurboIntegrator Parameters

These parameters are defined by the TurboIntegrator process and must be of the correct type (number or string).

Parameter	Description Value/Required/Default
<ti_parm>	Provide the string or number value *<value>* to the parameter named *<ti_parm>* which must be a valid parameter name accepted by the TurboIntegrator being run. <value>/No/None

Password Parameters

Passwords are either provided in cleartext (not recommended) using the pwd parameter, or using an encrypted file provided by the `passwordfile` parameter.

Parameter	Value/Required/Default	Description
pwd	String/No/None	Cognos TM1 or CAM password
`passwordfile`	String/No/None	The full path of the file containing the encrypted password for the specified user. If no path is specified, the Cognos TM1 server directory is assumed. When this option is used, you cannot use `-pwd`.
`passwordkeyfile`	String/No/None	If `passwordfile` is set, the full path to the key file is also required in order to decrypt the password. The password file and key file can be created using TM1Crypt tool. Refer to *IBM Planning Analytics Installation and Configuration*.

TM1RunTI configuration file

TM1RunTI can function with or without a configuration file.

If a configuration file is specified, its parameters are read first.

Parameters specified on the command line are then used to override those obtained from the configuration file. When a configuration file is read, TM1RunTI first obtains parameters from the [TM1RunTI] section of the configuration file.

If a connect parameter is present, then parameter values are obtained from the associated [Connect <name>] section and used to override anything read from [TM1RunTI].

A `-connect` parameter can also be provided on the command line, and overrides any connect parameters found in the configuration file.

The configuration file contains:

1. A single TM1RunTI section.
2. One or more sections defining the TurboIntegrator processes that may be run.
3. Zero or more sections defining connection parameters.

All entries must start at column 1. Lines beginning with # are treated as comments.

Section names must be enclosed in square brackets [ ]. If a section name is repeated, only the first one is used.

Parameters within a section:

- cannot have blank lines between them
- can appear in any order
- are specified in keyword=value format.

Parameter values need to be enclosed in quotes (") if they contain whitespace.

Connect sections

To facilitate easy maintenance for different server environments such as development, test and production, connection parameters for each environment can be specified in separate sections. Each section is named using the prefix "Connect -" followed by a user defined name. For example:

```
[Connect - Production]

[Connect - Test]

[Connect - Development]
```

Process sections

Multiple process sections are permitted. Each section is named to match a process in the server.

Each TurboIntegrator process section is used to define the parameters of the TurboIntegrator process and their default values.

If there are multiple process sections with the same name, only the first one is used.

Example configuration file

This example shows the [TM1RunTI] section and a section for a single TurboIntegrator process ("my_ti_process"). The parameters and their default values, which may be overridden by parameters provided on the command line, are defined below each section header.

```
[TM1RunTI]
process=my_ti_process
connect=Production

[Process - my_ti_process]
num1="value1"
stringX="value2"
stringY="value3"

[Connect - Production]
adminhost=
server=MyTM1server
user="MyTM1AdminServer"
passwordfile="c:\tm1_admin_area\passwords\tm1_password.txt"
AdminSvrSSLCertAuthority=.\ssl\applixca.pem
AdminSvrSSLCertID=tm1adminserver
AdminSvrSSLCertRevList=
CAMNamespace=LOCAL_NTLM
```

Processing logic

Configuration parameters and command line parameters are processed in the following fashion:

1. If specified by -i, the configuration file is opened and any connect option specified in [TM1RunTI] is processed first.
2. Any other parameters in [TM1RunTI] are then processed and may override those specified by the connect parameter.
3. The command line parameter `-connect` is processed next, if present. It loads values from the associated [Connect - <connection_name>] section of the config file, overriding any values loaded by the preceding steps.
4. The remaining command line parameters are processed.

For example, if you save the configuration file in the preceding example with the name tm1tools.config and then you execute the following:

```
tm1runti -i ".\tm1tools.config" -passwordkeyfile c:\keystore\prodkey.dat -connect
prodsystem
```

Since the -i parameter was provided, the tool would do the following:

1. Open the config file and load the [tm1runti] section
2. Upon seeing the connect parameter in [tm1runti], load the parameter values from [Connect - `testsystem`]
3. Process the command line parameters:
   a. Upon seeing the connect parameter, load the parameters from [Connect – `prodsystem`]
   b. Replace the value for `passwordkeyfile`.

Configuration filename and location

The command line parameter `-i` can be used to specify a configuration filename. This is particularly useful if several Cognos TM1 servers are supported in the environment, as a different configuration file can be used for each server and like-named processes in different servers can be defined with different parameters.

TM1RuntTI return codes and error messages

The following error messages are used by TM1RunTI.

Return codes and error messages

Return Code
Message: Description

0
None: The program completed successfully.

1
Password not specified: Password not specified as an argument or as a password file.
Short Help text: Necessary parameters not provided (user, server, process). The short help is sent to stdout. Equivalent to `-?`
Invalid number of parameters at *<n>*: More parameters were detected than are actually supported by the program, beginning at the *<n>* parameter.

2
Server connection failed: The program was unable to make a connection to a Cognos TM1 server.

3
Calling process*<TI_name>* completed with minor errors: The TurboIntegrator process completed but with minor errors.

4
Calling process *<TI_name>* completed with messages:: The TurboIntegrator process completed but returned messages.

5
Error retrieving password: The program was unable to get the password from the password file. One of the other error messages listed may appear in `stderr` before this one, indicating more precisely the nature of the problem.

- **NULL key returned from reading <*filename*> key path**.
- **NULL password returned from reading <*filename*> password file**.
- **Error obtaining file status of <*filename*>.**
- **Error opening <*filename*>**.
- **Unable to allocate data for key**.
- **Error reading <*filename*> key file**.

6
TI process: *<TI_name>* not found on server: *<server_name>*: The TI process was not found on the specified server.

7
TI process: *<TI_name>* parameter cannot be read: Cannot read parameter information from the TurboIntegrator process.

8
TI process: *<TI_name>* no read access: Specified user does not have read access to the TurboIntegrator process.

9
calling process: *<TI_name>* called ProcessQuit.: The TurboIntegrator process called ProcessQuit.

10
calling process: *<TI_name>* aborted.: The TurboIntegrator process was aborted.

11
TI process: *<TI_name>* reading numeric parameter *<param_name>=<param_value>* failed: A non-numeric value was passed into a numeric TurboIntegrator parameter.

99
Other TI error: The TurboIntegrator process completed with an unspecified error.

Errors are also returned from the TM1API. They are displayed as (TM1 API Error) *<xxx>* where <xxx> is the value defined in the TM1API.

Modes of execution and error handling limitations

TM1RunTI can be run as a standalone executable, from within an operating system batch script, or from within a Cognos TM1 TurboIntegrator process.

The most straightforward way to run TM1RunTI from within TurboIntegrator is to use the ExecuteCommand() call to directly execute it. For example:

```
ExecuteCommand("tm1runti –i myconfig.config -connect prodserver –process update")
```

The ability to define connection and other relatively static parameters in a configuration file makes it possible to simplify the parameter list passed to TM1RunTI from a calling TurboIntegrator process, and to reduce maintenance effort by centralizing connection information.

Executing TM1RunTI directly from within a TurboIntegrator process using ExecuteCommand() has an important limitation. TM1RunTI returns an error code if it fails, but the ExecuteCommand() does not return the error code and there is no other mechanism in TurboIntegrator to access the return code after the call.

Another limitation to consider is that the process will have the same current drive and directory as the calling process (the server) which will be the database directory. This is documented in "TurboIntegrator Functions" on page 2.

To deal with errors, execute TM1RunTI from a batch script called by ExecuteCommand so that the error return code can be obtained in CMD.EXE through the ERRORLEVEL variable and so that error messages can be logged or intercepted by redirecting stderr. Various options are then available to the application designer for handling the error, such as:

- Write the error information to database.
- Write the error information to a file and then, in a subsequent TurboIntegrator process, load the information into a Cognos TM1 Cube. The cube can then be used for reporting, alerts, etc.

  Note: In versions 9.5.1 and earlier, this could create additional lock contentions.
- Write the error information to a file or files and then, within the calling TurboIntegrator process, use the FileExists() TurboIntegrator process function to test for the existence of that file or files. The process can then take conditional actions based on the existence of the files generated by the batch script.

TM1RunTI other considerations

These are some additional considerations when using TM1RunTI.

Password Security

The use of passwords on the command line for this utility is not recommended for production deployments. Instead of using passwords on the command line, the password should be passed to the program using the passwordfile parameter to specify a file that contains the encrypted password. A keyfile is also needed, to decrypt the password, and this is provided through the passwordkeyfile parameter. These files can be stored in a location accessible to the username running the tool, but under operating system protection so that other users cannot access them.

A combination of password and key can be generated by using TM1Crypt tool which comes with the standard Cognos TM1 installation. See *IBM Planning Analytics Installation and Configuration* for details.

Platform Portability

The TM1RunTI executable file name (tm1runti.exe) is all lowercase for platform portability and for consistency with tm1top and other Cognos TM1 server tools.

Serializing TurboIntegrator processes using synchronized()

IBM Cognos TM1 TurboIntegrator (TI) function called synchronized() can be used in a TurboIntegrator script to force serial execution of a designated set of TurboIntegrator processes.

Cognos TM1 application developers can define TurboIntegrator (TI) processes that execute in response to user actions or run as batch processes. Unless explicitly prevented from doing so, TurboIntegrator processes may execute in parallel. In some applications, TurboIntegrator processes should be serialized in order to improve performance efficiency. Prior to the introduction of this new function, application designers used various techniques to ensure that TurboIntegrator processes were serialized.

One technique is to rely on object locks to force serialization of the processes. Typically, a status value is written to a cube to invoke the cube's lock as it prepares for exclusive access mode. However, the introduction of Parallel Interaction (PI) may cause this method to fail. Normally, data writers conflict with other data writers. In this way, an executing TurboIntegrator process in a cube is either able to acquire the lock and run to completion, or it must wait until the lock is available. In PI mode, multi-version concurrency control allows multiple writers to perform their writes immediately.

Since this technique is no longer valid with PI enabled, synchronized() is available to explicitly invoke serialization in TurboIntegrator process code.

See the "Process Control TurboIntegrator Functions" section of the TurboIntegrator Functions chapter of the *IBM TM1 Reference* for details on using this function.

synchronized()

IBM Cognos TM1 TurboIntegrator (TI) function called synchronized() can be 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.

`lockName`

Value=String

Required?=Yes

Default-none

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.

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.).

Lock object maintenance and naming

Lock objects are managed internally by Cognos TM1 . No explicit creation or deletion is required of the user. Simply specify a lock object by name in a synchronized() call.

Lock object names are insensitive to case or embedded blanks. For example, if there is a lock object with the name 'Abc Def', that lock object can be referred to using the names 'ABCDEF', 'ab cd ef' etc. In other words, the execution of a TurboIntegrator process with a call to `synchronized( ‘Abc Def' )` will serialize with the execution of a process with a call to `synchronized( ‘ABCDEF' )`. Lock object names may not exceed 1023 characters in length.

Order of execution

A group of TurboIntegrator processes containing synchronized() calls to the same lock object are prevented from concurrently executing. However, their actual order of execution is unaffected. As long as they do not execute concurrently, the order in which they execute is determined by many other factors, including application design and operating system level scheduling. If order of execution is important, for example, if one TurboIntegrator process is dependent on updates made by another process, then it is up to the application designer to use other methods to ensure the desired order of execution.

MaximumTIObjectLocks configuration parameter

The MaximumTILockObjects parameter restricts the size of the object locked list. See *IBM Planning Analytics Installation and Configuration* for more information.

TurboIntegrator security is assigned by administrator

The admin who creates a TurboIntegrator process assigns the security privileges to the TurboIntegrator process.

A TurboIntegrator process can be created only by an administrator, who has the Admin privileges required to create a process. The administrator can assign rights to the process. The TurboIntegrator process has those rights regardless of the rights assigned to any user running the process.

Non-admin users need to have Read access to a TurboIntegrator processes in order to see the process in the interface and to execute the process. But the TurboIntegrator process itself retains the rights assigned by the administrator.

For example, consider a user and an administrator where:

- User U1 has only Read access to cube_1.
- The administrator creates a TurboIntegrator process that does a CellPutN into cube_1, which requires Write access to the cube.
- The administrator gives U1 Read access to the TurboIntegrator process.
- U1 can run this TurboIntegrator process and it will do the CellPutN even though the user only has Read access to cube_1. The same result is obtained if U1 has None access to cube_1.
- A user with only Read access to a TurboIntegrator process can only view and execute the process. The user can't edit the process to change the value being sent or the location where data is being put.
- The conditions described above are also true when a user executes a TurboIntegrator process from within a chore.

To prevent U1 from being able to access this TurboIntegrator process, the IBM Cognos TM1 administrator should not give U1 Read access to the TurboIntegrator process.

Chapter 8. Scheduling a Process for Automatic Execution with Chores

You can execute processes on demand and you can create a *chore* to execute processes at defined intervals. These two methods of execution are not mutually exclusive. You can execute any process on demand at any time, even if the process is scheduled for automatic execution as a chore.

A chore is the TM1 object that executes one or more processes at a user-defined frequency. A chore is comprised of:

- A list of processes to be executed.
- A start date and time for the initial execution of the chore.
- A frequency at which the chore is subsequently executed.

Once defined, chores can be activated and deactivated as required.

Access to chores functionality is controlled by user group security privileges. You must be a member of the ADMIN or DataAdmin group to create chores on a server. Users must have Read privilege to a chore to be able to view the chore in the Server Explorer and to manually execute the chore.

You can schedule a process for automatic execution as a chore from within TurboIntegrator.

Procedure

1. Click the **Schedule** tab in the TurboIntegrator window.
2. Select the **Schedule this Process as a Chore Named** option.
3. Enter a name for the process in the adjacent field. By default TurboIntegrator assigns the name of the process to the chore.
4. Click a date on the calendar to specify a start date for the initial execution of the chore.
5. Enter a Time to specify the start time for the initial execution of the chore.
6. Set the fields in the Chore Execution Frequency box to define the interval at which the chore is executed.
7. Select a **Run Chore Time** option.
   - **Local Server Time** - Runs at the local server time, including during Daylight Saving Time/Summer Time periods.
   - **UTC Time** - Always runs at UTC, regardless of local Daylight Saving Time/Summer Time.
8. Choose **File**, **Save** to save the process with the scheduling information.

   When you schedule a process from within TurboIntegrator, the chore is automatically activated and will be executed at the specified start time.

   You can also create a chore for a process (or a collection of processes) directly from the Server Explorer.
9. In the Server Explorer, select the **Chores** icon beneath the server on which you want to create the chore.
10. Choose **Chores**, **Create New Chore**.

    The Chore Setup Wizard opens.
11. In the Available list, select the process for which you want to create a chore.
12. Click the right arrow icon.
13. Click **Next**.
14. Click a date on the calendar to specify a start date for the initial execution of the chore.
15. Enter a time to specify the start time for the initial execution of the chore.
16. Set the fields in the Chore Execution Frequency box to define the interval at which the chore is executed.
17. Select a **Run Chore Time** option.
    - **Local Server Time** - Runs at the local server time, including during Daylight Saving Time/Summer Time periods.
    - **UTC Time** - Always runs at UTC, regardless of local Daylight Saving Time/Summer Time.
18. Fill the **Chore Schedule is Active** box.

19. Click **Finish**.

    The Save Chore As dialog box opens.

20. Enter a name for the chore and click **Save**.

Editing a Chore

To open a chore for editing in the Chore Setup Wizard:

Procedure

1. Select the chore in the left pane of the Server Explorer.
2. Choose **Chore**, **Edit Chore**.

Activating a Chore

To activate a chore that is currently deactivated:

Procedure

1. Select the chore in the left pane of the Server Explorer.
2. Toggle the **Chore**, **Activate** option on.

Deactivating a Chore

To suspend the regularly scheduled execution of a chore:

Procedure

1. Select the chore in the left pane of the Server Explorer.
2. Toggle the **Chore**, **Activate** option off.

Deleting a Chore

To delete a chore:

Procedure

1. Select the chore in the left pane of the Server Explorer.
2. Choose **Chore**, **Delete**.

   Note: You cannot delete an active chore. You must deactivate a chore before you can successfully delete it.

Executing a Chore on Demand

To execute a chore on demand:

Procedure

1. Select the chore in the left pane of the Server Explorer.
2. Choose **Chore**, **Execute**.

Using Chore Commit

ChoreCommit is a property of a chore that allows you to specify if the processes in a chore will be committed as a single transaction or if the processes in the chore are committed as multiple transactions.

A Chore executes a sequence of TurboIntegrator processes as a single Commit transaction. Any locks acquired by the first process are kept until the last process is complete. This means locks can be held for very long periods of time. ChoreCommit enables a Chore to optionally execute such that each TurboIntegrator process is committed as a transaction when the process is complete. Locks are then held only for the duration of a single process instead of for the length of the chore.

Chore Property

When setting up a chore, Chores can be identified as:

- Single Commit Mode

  All processes are committed as a single transaction. This is legacy and default behavior.
- Multiple Commit Mode

  Any processes that need to be committed do so as they are processed.

This property can be modified only when a Chore is inactive.

Running a chore at server startup

You can designate a chore as a "startup" chore that is processed when the server starts up.

To indicate that a chore should be run when the server starts up, use the configuration parameter StartupChores to identify a list of chores to be run before the server starts up. A chore is a set of tasks that can be executed in sequence that are typically TurboIntegrator processes. See *IBM Planning Analytics Installation and Configuration* for information on this parameter.

Startup chores can be used as a way to set up the server before processing. Startup chores run before users login and before other chores begin processing.

Since Startup chores are run before logins are allowed, the user can not monitor the Startup chores with TM1Top and therefore there is no way to cancel a Startup chore with the exception of killing the server process.

Chapter 9. Java extensions support in TurboIntegrator

You can use Java™ as a scripting language to create and run TurboIntegrator processes.

Most TurboIntegrator functions are supported in Java and can be used like any other Java method. You can create TurboIntegrator processes with Java and use Java libraries to expand the capabilities of IBM Cognos TM1.

Java extensions are not supported in TM1 Rules.

Enabling and configuring Java extensions

You must complete the following procedures to enable and configure Java extensions in IBM Cognos TM1.

Configure the TM1 server to support Java

To enable Java support in TurboIntegrator, you must specify the location of the Java virtual machine .dll (jvm.dll) in your tm1s.cfg file. If you are running a 64-bit version of tm1s.exe, you must specify a 64-bit version of jvm.dll. A 32-bit version of tm1s.exe requires a 32-bit version of jvm.dll.

Procedure

1. In a text editor, open tm1s.cfg for the TM1 server on which you want to enable Java.

   The tm1s.cfg file is in the TM1 server's data directory. You can use the IBM® Cognos® Configuration tool to confirm the location of the tm1s.cfg file for your server.
2. Add the line `JavaJVMPath=<full_path_to_jvm.dll>` to the tm1s.cfg file.

   (On UNIX, the path would be the equivalent to `jvm.dll`, for example, `libjvm.so`).
3. Save the tm1s.cfg file.

   JavaJVMPath is a dynamic configuration parameter and does not require a restart of your TM1 server.

Create Java extensions directories

You must create directories to store your Java extensions.

About this task

There are several specific directories in which TM1 searches for extensions, each intended to store a particular category of extensions. Depending on your implementation, you might need to create any combination of these directories.

Procedure

1. To store IBM-approved extensions, create the `\javaextensions\ibm` subdirectory of the `<tm1_install_dir>\bin64` directory.

   The default 64-bit TM1 installation directory is `C:\Program Files\IBM\cognos\tm1_64`, so the full path for the directory where IBM-approved extensions are stored would be `C:\Program Files\IBM\cognos\tm1_64\bin64\javaextensions\ibm`.
2. To store install-wide user extensions, create the `\javaextensions\user` subdirectory of the `<tm1_install_dir>\bin64` directory.
3. To store model-specific extensions, create the `}javaextensions\ibm` subdirectory of your TM1 server's data directory.
4. To store user-specific extensions, create the `}javaextensions\user` subdirectory of your TM1 server's data directory.
5. After you create the subdirectories, restart your TM1 server.

Creating the javaextensions.policy file

The `javaextensions.policy` file, which must be located in the `<tm1_install_dir>\configuration` directory, allows Java extensions to execute certain specified classes.

Procedure

1. In the `<tm1_install_dir>\configuration` directory, create a text file named `javaextensions.policy`.
2. To allow all classes to be executed from java extensions, insert the following statements: `grant { permission java.security.AllPermission; };`
3. Save the `javaextensions.policy` file.

Configure Eclipse Java EE IDE

Follow these steps to configure Eclipse Java EE IDE for IBM Cognos TM1 Java extensions.

About this task

This procedure and the children topics of this procedure guide you through the configuration of Eclipse and the creation of simple example scripts that illustrate the use of Java extensions in TM1. You can modify the examples to create your own custom scripts.

Procedure

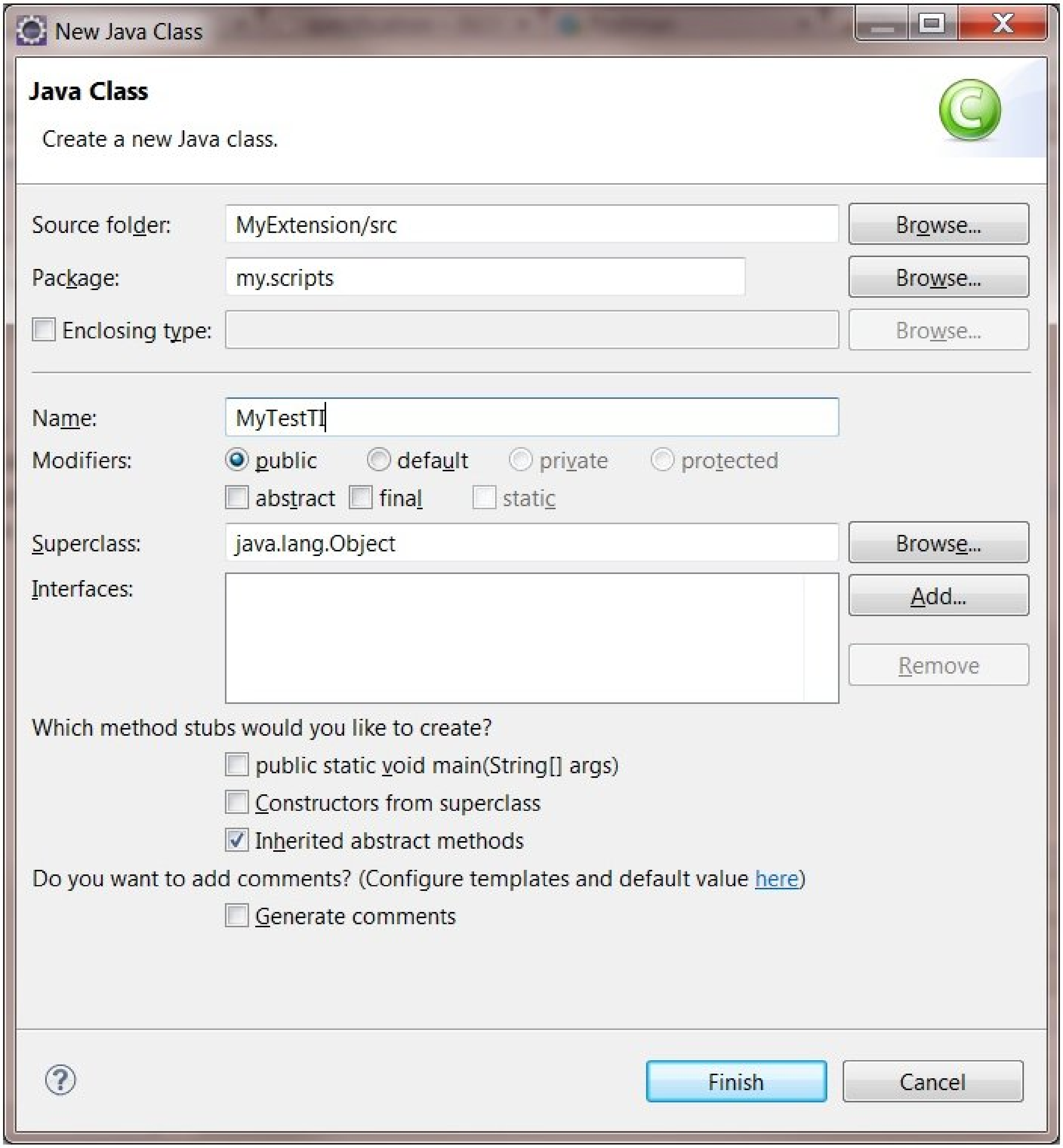
1. Open Eclipse.
2. Click **Workbench**.
3. Click **Window** > **Open Perspective** > **Java** to open the Java perspective.
4. Click **File** > **Java Project** to create a new Java project.
5. Enter a **Project name**.
6. Click **Finish**.
7. Right-click the new project, then click **Properties**.
8. On the **Properties** window, click **Java Build Path**.
9. Click the **Libraries** tab.
10. Click **Add External JARs**.
11. Add the `javatiapi.jar` from your TM1 installation directory.
12. Click **OK** to save the properties.
13. Right-click the new project, then click **New** > **File**.
14. Enter `extension.xml` as the file name.
15. Click **Finish**.
16. Click the **Source** tab, then enter the following XML code:

    ```
    <extension>
        <id>my.extension</id>
        <name>My Extension</name>
        <version>1.0.0.0</version>
        <scripts-package>my.scripts</scripts-package>
    </extension>
    ```

    This XML tells the Java extensions code that this is an extension, and tells it where to look for these items:

    - id - An unique ID for your extension, each extension must have a different ID.
    - name - A display name for your extension, possibly a brief description
    - version - The version number for your extension. Java extensions will always load the latest version of an extension if several extensions are all available with the same ID.
    - scripts-package - Tells the Java extensions engine where to look for Java extensions TurboIntegrator scripts.
17. Expand the new project in Eclipse.
18. Right-click the SRC folder, then click **New** > **Package**.

19.Assign the name `my.scripts` to the new package, then click **Finish**.
20.In the **Package** pane, right-click the new `my.scripts` package, then click **New** > **Class**.
21.Complete the New Java Class dialog box as follows:



Source folder: MyExtension/src
Package: my.scripts
[ ] Enclosing type:
Name: MyTestTI
Modifiers: [✓ public] default private protected
[ ] abstract [ ] final [ ] static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
[ ] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

Figure 2: New Java Class dialog box

22.Click **Finish**.
23.Enter the following code in the `MyTestTI.java` class:

```
package my.scripts;
```

```
import com.ibm.cognos.tm1.javati.TM1UserThread;
import com.ibm.cognos.tm1.javati.ti.TIFunctions;

public class MyTestTI {
    public static double MyTestTI(String[] args) {
        TIFunctions ti = TM1UserThread.getTIFunctions();
        if (args.length < 1) {
            throw new IllegalArgumentException();
        }
        String dim = args[0];
        if (ti.DimensionExists(dim)) {
            ti.DimensionDeleteAllElements(dim);
        } else {
            ti.DimensionCreate(dim);
        }
        return 0;
    }
}
```

This script takes a single parameter, the name of a dimension. If that dimension exists it deletes all elements, otherwise it creates a new dimension with that name. It's a simple script but it is a useful example.

Create an Ant build file

After you write a class, you need to compile it into a JAR file in the correct location. The easiest way to do this is to create an Ant build file.

Procedure

1. Right-click the `MyExtension` project, then click **New** > **File**.
2. Specify `MyExtension` as the parent folder and `buildjar.xml` as the **File name**, then click **Finish**.
3. Enter the following code in the new `buildjar.xml` file:

```
<project name="MyExtension" default="createjar">
    <property name="projectHome" location="." />
    <property name="targetdir" location="C:/TM1/Models/PlanSamp/}javaextensions/
user" />
        <target name="createjar">
            <jar destfile="${targetdir}/myextension.jar" basedir="${projectHome}/
bin">
                <manifest>
                    <attribute name="Class-Path" value="."/>
                </manifest>
            <fileset file="${projectHome}/extension.xml"/>
            </jar>
        </target>
    </project>
```

Set the `targetdir` property location to the full path to the `}EmbeddedJava` folder in your TM1 server data directory.

Now you can set up Eclipse so that every time you change a file in the project it rebuilds your JAR file automatically.

4. Right-click the `MyExtension` project, click **Properties**, and then click the **Builders** category.
5. Click **New** on the Properties for MyExtension window.
6. Click **Ant Builder**, then click **OK**.
7. On the Edit Configuration window, click **Browse Workspace** under **Buildfile**, then navigate to your Ant build file.
8. Select your `buildjar.xml` file, then click **OK**.
9. Click the **Targets** tab on the Edit Configuration window.
10. Click **Set Targets** next to the **Auto Build** field.
11. Select the only available target (`createjar`), then click **OK**.
12. Click **OK** on all subsequent prompts and dialog boxes.

Your JAR builds immediately and automatically rebuilds every time that you change any of your source files.

Create a wrapper process to run Java extensions scripts in TurboIntegrator
After you configure Eclipse, you must create a wrapper process to run TurboIntegrator script from Architect.

Procedure

1. Open IBM Cognos TM1 Architect and log on to the server that is configured to run Java extensions.
2. Create a TurboIntegrator process.
3. Enter the following statement in the Prolog tab of the new process:
   `ExecuteJavaN('my.scripts.MyTestTI', 'Hello');`
4. Save and execute the process.

Results

This example creates a new dimension named `Hello`.

The first parameter to the ExecuteJavaN function is the fully qualified name of the Java extensions class that you created earlier. The rest of the parameters that are passed to ExecuteJavaN are passed as arguments to the Java extension.

Java Extensions TurboIntegrator functions

The ExecuteJavaN and ExecuteJavaS functions let you execute Java scripts from a TurboIntegrator process.

ExecuteJavaN

ExecuteJavaN executes a Java TurboIntegrator process that returns a number. If you want to execute a Java TurboIntegrator process that returns a string, use ExecuteJavaS.

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

Syntax

```
ExecuteJavaN('JavaTIClass', ['OptionalParameter1', 'OptionalParameter2', ...] )
```

Argument	Description
JavaTIClass	The fully qualified name of the Java TurboIntegrator class you want to execute.
OptionalParameters	Optional parameters that are passed to the Java TurboIntegrator process itself. You can pass as many parameters as necessary, including none. You can pass only strings as parameters, you cannot pass numbers. You can use the StringToNumber TurboIntegrator function to pass numbers to Java TurboIntegrator scripts. For more information, see "StringToNumber" in the *TM1 Reference*.

A Java TurboIntegrator class, which returns a number and can be called from ExecuteJavaN, must use the following pattern:

```
package com.example;

import com.ibm.cognos.tm1.javati.JavaTI;

@JavaTI
```

```
public class MyTestTI {
    public static double MyTestTI (String [] args) [
    ...
    return ...;
    }
}
```

Example

`ExecuteJavaN('com.example.MyTestTI', 'First', 'Second', 'Third');`

ExecuteJavaS

ExecuteJavaS executes a Java TurboIntegrator process that returns a string. If you want to execute a Java TurboIntegrator process that returns a number, use ExecuteJavaN.

This is a TM1 TurboIntegrator function, valid only in TM1 TurboIntegrator processes.

Syntax

```
ExecuteJavaS('JavaTIClass', ['OptionalParameter1', 'OptionalParameter2', ...] )
```

Argument	**Description**
JavaTIClass	The fully qualified name of the Java TurboIntegrator class you want to execute.
OptionalParameters	Optional parameters that are passed to the Java TurboIntegrator process itself. You can pass as many parameters as necessary, including none. You can pass only strings as parameters, you cannot pass numbers. You can use the StringToNumber TurboIntegrator function to pass numbers to Java TurboIntegrator scripts. For more information, see "StringToNumber" in the *TM1 Reference*.

A Java TurboIntegrator class, which returns a string and is called from ExecuteJavaS, must use the following pattern.

```
package com.example;

import com.ibm.cognos.tm1.javati.JavaTI;

@JavaTI
public class MyTestTI {
    public static String MyTestTI (String [] args) [
    ...
    return ...;
    }
}
```

Example

`ExecuteJavaS('com.example.MyTestTI', 'First', 'Second', 'Third');`

Support for arrays in functions that accept a variable number of parameters

One advantage that Java extensions in TurboIntegrator have over normal TurboIntegrator scripts is that Java extensions support passing a variable number of parameters as an array.

Several TurboIntegrator functions support a variable number of parameters, including the CubeCreate, CellGetN, CellGetS, CellPutN, and CellPutS functions.

For example, the CubeCreate function takes one parameter for the name of the new cube to create and a variable number of parameters for the dimensions that comprise the cube. You can pass the dimension parameters directly, as in this example:

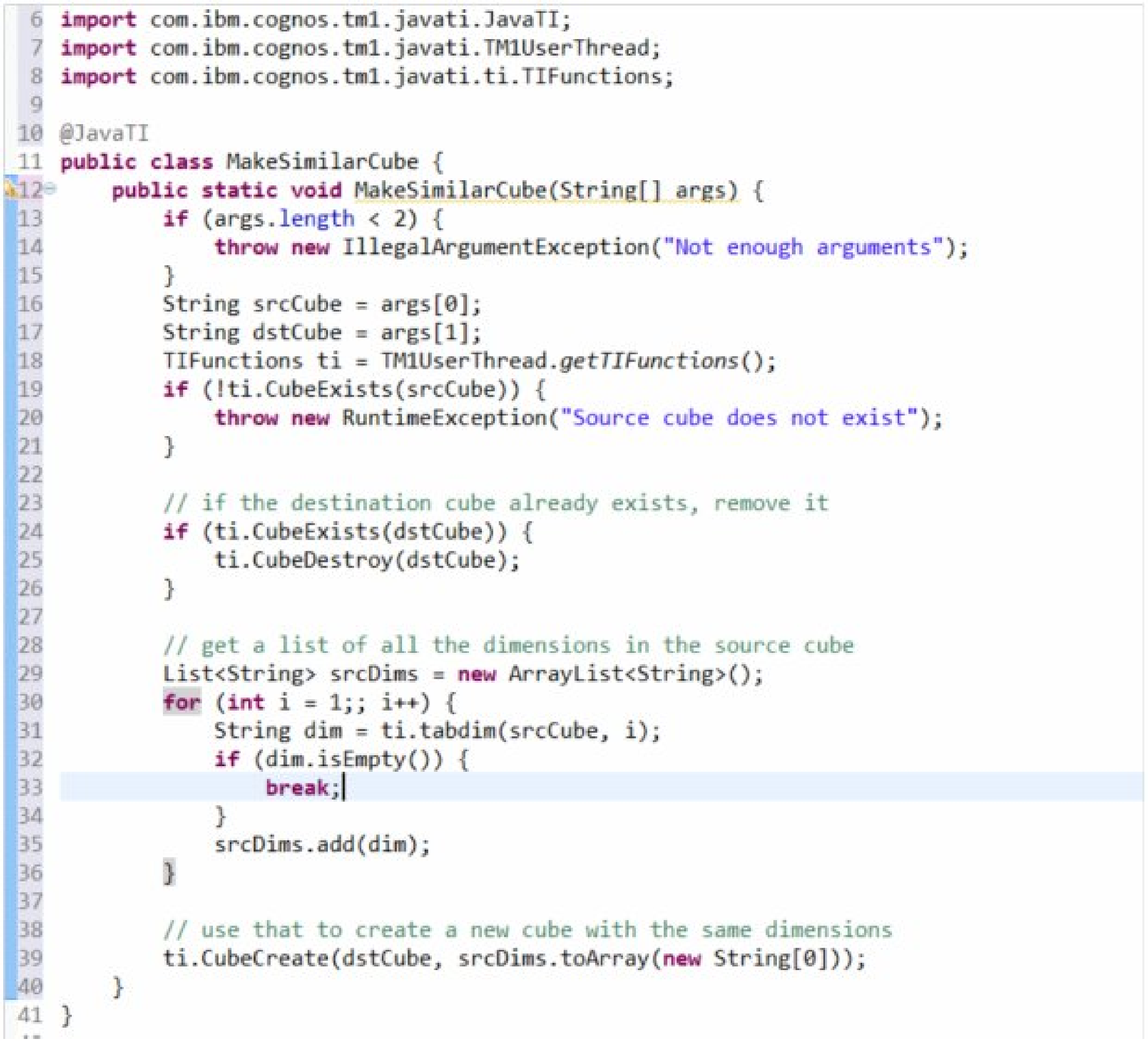
```
ti.CubeCreate("Sales", "Months", "Regions", "Versions", "Sales Metrics");
```

Or you can pass an array of values to the function, as in this example:

```
String[] dims = new String[] {"Months", "Regions", "Versions", "Sales Metrics"};
ti.CubeCreate("Sales", dims);
```

Because you can pass an array of values, you can pass any number of parameters. You don't have to decide when you write the script how many parameters a function must take. When you run the script you can create an array of any size, and pass it to a function (such as CubeCreate) that accepts a variable number of parameters and it works as expected.

To demonstrate this capability, here is an example of a Java TurboIntegrator script that can create a new cube that has the same dimensionality as an existing cube. This script works irrespective of the size of the existing cube.



```
import com.ibm.cognos.tm1.javati.JavaTI;
import com.ibm.cognos.tm1.javati.TM1UserThread;
import com.ibm.cognos.tm1.javati.ti.TIFunctions;

@JavaTI
public class MakeSimilarCube {
    public static void MakeSimilarCube(String[] args) {
        if (args.length < 2) {
            throw new IllegalArgumentException("Not enough arguments");
        }
        String srcCube = args[0];
        String dstCube = args[1];
        TIFunctions ti = TM1UserThread.getTIFunctions();
        if (!ti.CubeExists(srcCube)) {
            throw new RuntimeException("Source cube does not exist");
        }

        // if the destination cube already exists, remove it
        if (ti.CubeExists(dstCube)) {
            ti.CubeDestroy(dstCube);
        }

        // get a list of all the dimensions in the source cube
        List<String> srcDims = new ArrayList<String>();
        for (int i = 1;; i++) {
            String dim = ti.tabdim(srcCube, i);
            if (dim.isEmpty()) {
                break;
            }
            srcDims.add(dim);
        }

        // use that to create a new cube with the same dimensions
        ti.CubeCreate(dstCube, srcDims.toArray(new String[0]));
    }
}
```

Figure 3: An example of a Java TurboIntegrator script

This example gets the dimensions of the existing cube as a list, convert that list to an array, and passes the array to the CubeCreate function.

TurboIntegrator functions not supported in Java Extensions

The following TurboIntegrator functions are not supported in Java Extensions. TurboIntegrator functions not appearing on this list are supported.

- ASCIIDelete
- ASCIIOutput
- SetInputCharacterSet
- SetOutputCharacterSet
- SetOutputEscapeDoubleQuote
- TextOutputFormatDate

- NewDateFormatter
- ParseDate
- DimensionTopElementName
- ODBCCLose
- ODBCOpen
- ODBCOPENx
- ODBCOutput
- SetODBCUnicodeInterface
- ProcessError ExecuteCommand
- ExecuteProcess
- GetProcessErrorFileDirectory
- GetProcessErrorFileName
- GetProcessName
- ItemReject
- ItemSkip
- ProcessBreak
- ProcessQuit
- Synchronized
- GetUseActiveSandboxProperty
- ServerActiveSandboxGet
- ServerActiveSandboxSet
- SetUseActiveSandboxProperty
- ServerSandboxExists
- ServerSandboxGet
- ServerSandboxListCountGet
- CellSecurityCubeCreate
- CellSecurityCubeDestroy
- PublishView
- DataSourceSAPUsingRoleAuths
- DataSourceSAPUsingTexts

Debugging Java extensions

You can debug IBM Cognos TM1 Java extension by writing errors and exceptions to the TM1 log file, as well as by using Eclipse to remotely debug your Java code.

Debugging Java code with Eclipse

It is possible to remotely debug your Java code by using Eclipse. Being able to debug processes by using a proper debugging tool is a major advantage that Java extensions have over regular TurboIntegrator scripts.

Procedure

1. Open the `Tm1s.cfg` file for the TM1 server where Java extensions are enabled.
2. Add the following parameter to the file:

   ```
   JavaJVMArgs=-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=1044
   ```

3. Save `Tm1s.cfg`.
4. Restart the TM1 server.
5. In Eclipse, click **Run** > **Debug Configurations**.
6. Scroll down the tree on the left pane. Locate and double-click **Remote Java Application** to create a new profile.

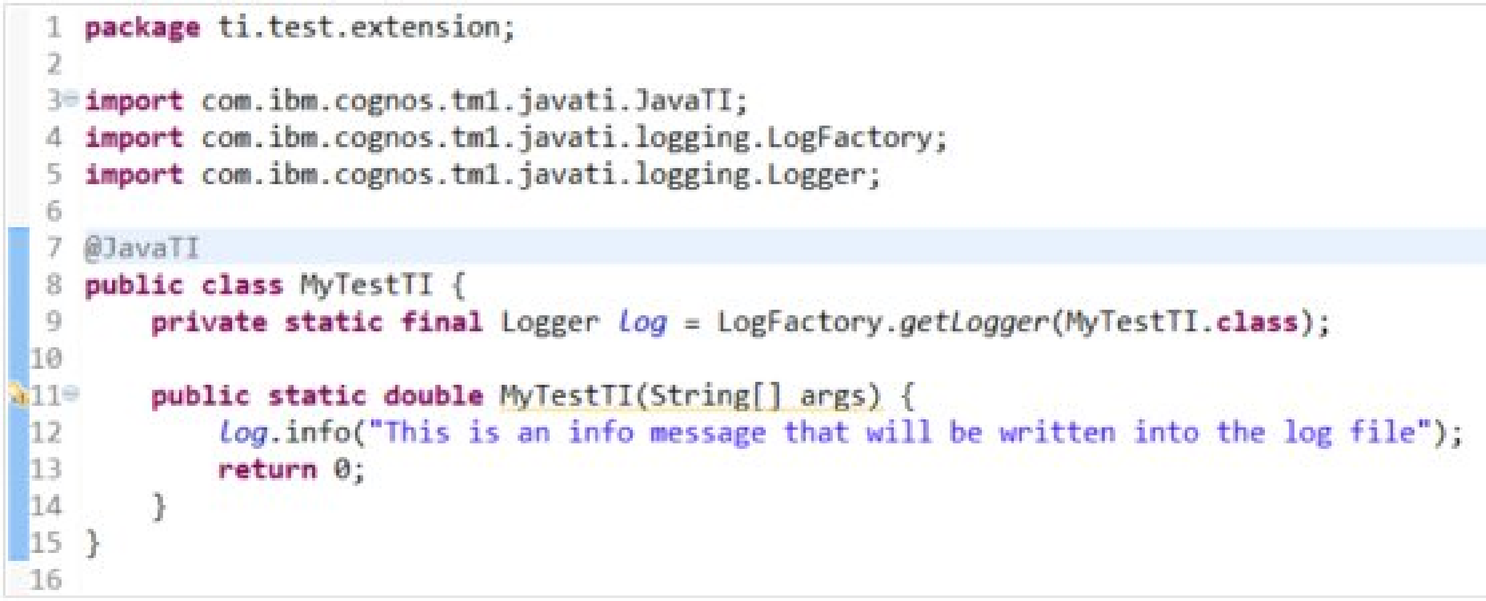
7. Change the default **Port** value to 1044. This corresponds to the address value in the JavaJVMArgs parameter you created in Step 2.
8. Click **Debug** to start a remote debugging session.
9. To add a breakpoint to your code, you can do one of the following:
   - Double-click in the margin of the line where you want to have a breakpoint.
   - Place the cursor in the line where you want a breakpoint and pressing **Ctrl+Shift+B**.
10. Re-run your wrapper TurboIntegrator process with Java extensions.

    The execution will break at your breakpoint and you can step through.

Writing errors and other information to the TM1 server log file

Java extensions in TM1 include the capability to write to the `tm1server.log` file from Java scripts, which allows you to record information and events that are associated with your scripts.

Here is an example of a script that writes an information message to the TM1 server log file:



```
package ti.test.extension;

import com.ibm.cognos.tm1.javati.JavaTI;
import com.ibm.cognos.tm1.javati.logging.LogFactory;
import com.ibm.cognos.tm1.javati.logging.Logger;

@JavaTI
public class MyTestTI {
    private static final Logger log = LogFactory.getLogger(MyTestTI.class);

    public static double MyTestTI(String[] args) {
        log.info("This is an info message that will be written into the log file");
        return 0;
    }
}

```

Figure 4: Java extension debugging script

The Logger class has several methods that can be used to write at different log levels:

- debug - a statement that is only shown when explicitly trying to debug something
- info - a log entry that provides useful (but non-essential) information
- warn - a log entry that describes some kind of non-critical issue
- error - a log entry to describe something that is a substantial error
- fatal - a log entry that describes a critical failure

Each of these functions can be used as in the example, taking a string that is written to the log file. Additionally, the functions can accept both a string and an exception, in which case the stack trace for the exception is also logged.

Integrating TM1 and Cognos Command Center using JavaTI

By combining the CCC REST API and TM1 Action Buttons, a TM1 user can initiate the run of a CCC process by clicking an action button.

About this task

The overall flow is:

- The TM1 Action Button starts a TI process,

- The TI process then makes a JavaTI call into the CCC TM1 wrapper,
- The CCC TM1 wrapper then makes an HTTPS REST request to the CCC server.

The CCC TM1 wrapper is distributed as a .jar file that contains the following JavaTI classes:

- `StartProcessAsync(apiEndpoint, username, password, ecosystem, environment, process, parameters)`

  Note: For the Async request to work you must set Request Queue Length to a number bigger than 0 (zero) in Tools > System Configuration > Event Listener. This defines the maximum number of concurrent requests for a given process.
- `StartProcess(apiEndpoint, username, password, ecosystem, environment, process, parameters)`

The arguments to these operations are:

- `apiEndpoint`: String, https://hostname:port For example: https://localhost:9004
- `username`: String, the username to authenticate
- `password`: String, the password to authenticate
- `ecosystem`: String, the name of the ecosystem
- `environment`: String, the name of the environment
- `process`: String, the name of the process
- `parameters`: Optional string containing name=value pairs for process parameters. Multiple pairs are separated with space and values containing spaces must be surrounded by double quotes, for example: param=value other_param="other value".

The return value for these operations is:

- 0 (false) = success
- 1 (true) = an error occurred

Procedure

1. In TM1®, configure the TM1 Server to support Java™:
   a) Edit tm1s.cfg.
   b) Add the line: JavaJVMPath=full_path_to_jvm.dll.

      The jvm.dll is typically in tm1_install_dir\bin64\jre\7.0\bin\default\jvm.dll
2. Create Java extensions directories (for install-wide extensions): Create the directory tm1_install_dir \bin64\javaextensions\user.
3. Create the javaextensions.policy file:
   a) In the tm1_install_dir\configuration directory, modify the text file named javaextensions.policy.
   b) To allow all classes to be executed from Java extensions, insert the following statements: `grant { permission java.security.AllPermission; }`
   c) Save the javaextensions.policy file.
4. Copy the Cognos® Command Center® Java extensions for Turbo Integrator jar file to the Java extensions directory created in step 2:
   a) Copy the .jar file from CCC INSTALLDIR\TM1wrapper to tm1_install_dir\bin64\javaextensions\user.
   b) Add CCC REST API certificate to TM1 Java JRE certificates.

      This can be done with keytool found in tm1_install_dir\bin64\jre\bin. For example, `tm1_install_dir \bin64\jre\7.0\bin\keytool.exe -importcert -alias CCCRESTAPI -file ccc_rest_api_certificate.cer -keystore "tm1_install_dir\bin64\java\jre\lib\security \cacerts" -storepass changeit`.
   c) Restart the TM1 server.
5. Create a TM1 TurboIntegrator process that calls the Cognos Command Center wrapper:
   a) Open IBM® Cognos TM1 Architect and log on to the server that is configured to run Java extensions.
   b) Create a new TurboIntegrator process.

c) Enter the following statement in the Prolog tab of the new process:

```
ExecuteJavaN('com.ibm.cognos.pmccsrvtm1wrapper.StartProcess', 'api-endpoint',
'user', 'password', 'Ecosystem', 'Environment', 'Process', 'parameters'),
```

Where `api-endpoint` is the URL of the CCC API, for example https://localhost:9004, `user` is the name of a CCC user that is running the process, `password` is the password for the user, `ecosystem` is the name of the ecosystem, `environment` is the name of the environment within the ecosystem, `process` is the name of the process within the ecosystem, and `parameters` is an optional string containing name=value pairs for process parameters. Multiple pairs are separated with space and values containing spaces must be surrounded by double quotes, for example: parm=value other_param="other value". The return value is an 0 of success. In case of a failure, the non-zero value is returned.

d) Save and execute the process.

6. Follow the steps in the topic Configuring an Action Button to Run a Process (http://www.ibm.com/support/knowledgecenter/SS9RXT_10.3.0/com.ibm.swg.ba.cognos.tm1_dg_dvlpr.10.3.0.doc/t_configurethebuttontorunaprocess_n50131.html%23ConfiguretheButtontoRunaProcess_N50131) to configure the TM1 Action Button to run the TM1 Turbo Integrator process that is created in step 5.

Appendix A. TurboIntegrator Tutorial

This tutorial guides you through advanced IBM Cognos TM1 TurboIntegrator features.

This tutorial is designed for users responsible for implementing TM1 and developing utilization strategies within their organization. The advanced user, or developer, will typically be responsible for creating, maintaining, and developing cubes and dimensions as well as data import processes. Before working through this tutorial, you should have a good understanding of TM1 concepts and a working knowledge of TM1 functionality.

The tutorial teaches you how to use TurboIntegrator to create dimensions and cubes, import flat files and ODBC data sources. It will also show you how to expand the power of TurboIntegrator using advanced scripting features. This tutorial also includes hints and tips on how to approach a TurboIntegrator problem.

Setting the Tutorial Data Directory

This tutorial relies on sample data shipped with TM1 . Before beginning this tutorial, you must set your local server data directory to reference the sample data.

To set your data directory:

Procedure

1. Click **TM1** in the left pane of the Server Explorer and select **File**, **Options**.

   The Options dialog box opens.
2. Click the **Browse** button for the Local Server Data Directory to navigate to the TurboIntegrator sample data directory.

   The sample data directory is named `TI_data`, and it is located in your `<`*`install_dir`*`>\samples\tm1\TI_Data\` directory, for example, `C:\Program Files\IBM\Cognos\TM1_64\samples\tm1\TI_Data\`.
3. Click **OK** on the Options dialog box to set the data directory and restart the local server.

TurboIntegrator Overview

TM1 TurboIntegrator lets you create processes that automate data importation, metadata management, and other tasks.

A process is an object that consists of:

- A description of a data source
- A set of variables corresponding to each column in the data source
- A set of maps that define relationships between variables and data structures in the TM1 database.
- A prolog procedure, consisting of a series of actions to be executed before the data source is processed.
- A metadata procedure, consisting of a series of actions that update or create cubes, dimensions, and other metadata structures.
- A data procedure, consisting of a series of actions to be executed for each record in the data source.
- An epilog procedure to be executed after the data source is processed.
- A set of parameters that can be used to generalize a process so it can be used in multiple situations.

You can use TurboIntegrator to import data from ODBC sources, ASCII files, SAP-based data, OLAP multi-dimensional sources, TM1 cube views, and TM1 dimension subsets.

TurboIntegrator includes a complete set of functions that you can use to enhance process capabilities. You can use these functions to create scripts that export data to ASCII files and ODBC sources, or that use conditional expressions to control processing. 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 and UNDEFVALS functions.

Access to TurboIntegrator is controlled user groups. You must be a member of the ADMIN group to gain access to all TurboIntegrator features and define processes on a networked TM1 server.

There is no interface to assist in the creation of TurboIntegrator functions. You must 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.

Creating a TurboIntegrator Process

There are five steps to creating a process. Each step is completed by setting options or editing values in an individual tab of the TurboIntegrator window.

The steps required to create a process include:

Procedure

1. Defining a data source
2. Setting variables
3. Mapping data
4. Editing advanced scripting
5. Scheduling the completed process

   You must complete each tab of the TurboIntegrator window in sequential order to create a process. TurboIntegrator does not let you proceed to a new tab until you supply all required information in the current tab.

Creating Dimensions Using TurboIntegrator

You can use TM1 TurboIntegrator to create a list of elements for a dimension from one of several possible data sources, including ODBC and ASCII files. This is the fast approach to creating a long list of elements, such as a thousand names along a customer dimension.

Sample ASCII File
Here is the delimited ASCII file (example.cma) that you will use to build a dimension and import data.

```
"New England", "Massachusetts", "Boston", "SuperMart",
"Feb" , 2000000"New England", "Massachusetts", "Springfield", "SuperMart",
"Feb" , 1400000"New England", "Massachusetts", "Worcester", "SuperMart",
"Feb" , 2200000
```

Each record in this source file has six fields, three of which will be used to create the Example dimension. The first two fields will become consolidated elements. The third field will become a numeric element. The remainder of the fields will be ignored.

In the Dimension Editor, the Example dimension will have the following structure:

New England

- Massachusetts
  - Boston
  - Springfield
  - Worcester

The numeric values from Boston, Springfield, and Worcester will consolidate into Massachusetts totals, which will consolidate into New England totals.

Creating a Dimension from an ASCII File
To create a dimension using the example.cma sample file:

Procedure

1. In the left pane of the Server Explorer, select **Processes** beneath the local server.
2. Choose **Process**, **Create New Process**.

   The TurboIntegrator window opens.
3. Select **Text** as the Data Source Type.
4. Click the **Browse** button for the Data Source Name and select **example.cma** in your TI_data directory.
5. Leave the Data Source Name on Server field empty.
6. Set the Delimiter Type to **Delimited** and the Delimiter to **Comma**.
7. Disregard the Quote Character and the Number of Title Records fields, as there are no quotes and no title records in the input file.

   The Decimal Separator should be period (.) and the Thousand Separator should be comma (,).
8. Click the **Preview** button to view records from the example.cma source file. These records let you examine the structure of records in the data source.

Identifying Variables
After loading the source data into TurboIntegrator, you must identify the contents of each field in the source. TM1 assigns a variable to each field in the source.

Procedure

1. Click the **Variables** tab to reveal the following information, which displays a row for each variable in the data source.

Variable Name	**Variable Type**	**Sample Value**	**Contents**
V1	String	New England	Ignore
Massachusetts	String	Massachusetts	Ignore
Boston	String	Boston	Ignore
Supermart	String	Supermart	Ignore
Feb	String	Feb	Ignore
V6	Numeric	2000000	Ignore

   The first column of the grid assigns a Variable Name to each data source field. To assign your own variables, click the appropriate cell and enter a new variable name.

   The second column assigns a Variable Type for each variable. This identifies the type of data in the source field. You can change the type by selecting one from the drop-down list.

   The third column, Sample Value, lists the contents of the first record of the data source. In the above image, New England is the contents of the first field in the first record of example.cma.

   The Contents column determines the data type (Element, Consolidation, Data, Attribute, Other, or Ignore) each variable identifies. In the example, the first three variables identify consolidations and elements of a regional hierarchy.
2. In the Contents column for variable V1, select **Consolidation** from the drop-down list.
3. Do the same for theMassachusetts variable.
4. For the Boston variable, select **Element**.
5. Select **Ignore** for all other variables, as they will not be used to create the dimension.

Variable Name	Variable Type	Sample Value	Contents
V1	String	New England	Consolidation
Massachusetts	String	Massachusetts	Consolidation
Boston	String	Boston	Element
Supermart	String	Supermart	Ignore
Feb	String	Feb	Ignore
V6	Numeric	2000000	Ignore

Mapping Variables
After identifying variables in the data source, you must map them to TM1 objects.

Procedure

1. Click the **Maps** tab, then the **Cube** sub-tab.
2. You are not creating a cube, so select **No Action** in the Cube Action box.
3. The Data Action is irrelevant, as you are not creating or updating a cube. You can disregard this box.
4. The Cube Logging option is not relevant, as you are not processing data values. Leave this option unselected.
5. Click the **Dimensions** sub-tab.

   This grid has a row for each variable that you identified as an Element content type. You must specify an element type and identify the dimension to which the element belongs.
6. You are creating a new dimension, so type **Example** in the Dimension column for the Boston variable.
7. Select **Create** from the Action drop-down list.
8. Select **Numeric** from the Element Type drop-down list.

   The Boston variable is now mapped as a numeric element of a new dimension named Example.

   You can now map the variables identified as consolidations.
9. Click the **Consolidations** sub-tab.

   TM1 correctly identifies both consolidation variables as members of the new Example dimension. All you have to do is identify the child variable for each consolidation.
10. For the **V1**Consolidation Variable, select **Massachusetts** as the Child Variable.
11. For the **Massachusetts** Consolidation Variable, select **Boston** as the Child Variable.
12. Do not edit the Weight of either consolidation variable.

Consolidated Variable	Dimension	Child Variable	Weight	Sample Value	Component Order
V1	Example	Mass.	1.000000	New England	By Input
Mass.	Example	Boston	1.000000	Massachusetts	By Input

All mapping is complete. If you want, you can click the Advanced tab and then click through the various sub-tabs to view the scripts generated by TurboIntegrator that create the new Example dimension and insert consolidations and elements. We will take a closer look at TurboIntegrator scripts later in this tutorial.

Saving and Executing the Process
To save and execute the process:

Procedure

1. Click the **Run** button ⚡.

   TM1 prompts you to save the process.
2. Save the process as create_Example_dimension.

   It is a good idea to save processes with descriptive names.

   After a few seconds, you should see a message box displaying a confirmation that the process executed successfully.
3. Close the TurboIntegrator window.
4. Open the Server Explorer.
5. Right-click the new Example dimension and select **Edit Dimension Structure**.

   The Example dimension opens in the Dimension Editor.
6. Click to sort the dimension members by hierarchy level.

   The Example dimension was successfully created. New England is a consolidated element containing Massachusetts (consolidated element), which in turn contains Boston, Springfield, and Worcester (numeric elements).

Creating a Dimension from an ODBC Source

This part of the tutorial guides you through the creation of a dimension from an ODBC data source. The procedure is very similar to creating a dimension from an ASCII file.

Defining the Data Source

Before continuing with the tutorial, you must add a Microsoft Access database as an ODBC data source to make it available to TurboIntegrator.

Procedure

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

   The procedure required to access this dialog box varies according to the version of Windows you are running. See the Windows online help for details.
2. On the DSN tab, click the **Add** button.

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

   The ODBC Access Setup dialog box opens.
4. Type **NewDB** in the Data Source Name field.
5. Click the **Select** button.

   The Select Database dialog box opens.
6. Navigate to you TI_Data directory and select **NewDB.mdb**.
7. Click **OK** to exit the Select Database dialog box.
8. Click **OK** to exit the ODBC Administrator dialog box.

   The NewDB Access database is now available as an ODBC source.

Querying the Data Source

To query the data source:

Procedure

1. From the Server Explorer, right-click the Processes icon and select **Create New Process**.

   The TurboIntegrator window opens.
2. Select **ODBC** as the Data Source Type.

3. Click the **Browse** button next to the Data Source Name field.
4. The ODBC Data Sources dialog box opens.
5. Select **NewDB** and click **OK**.

   NewDB.mdb has one table, ACCOUNT, which has 27 fields. You will write an SQL query to select information from six of them. All ODBC queries *must* use the SQL dialect of the underlying DBMS. The syntax of an MS Access query will be different from that of an Informix® query, an SQL Server query, and so on.

   To guarantee the correct syntax, you can first create the query using the querying facility of the underlying DBMS, then copy the query and paste it into the TurboIntegrator Query field.
6. In the Query field, type the following statement exactly as it opens:

   ```
   SELECT [ACCOUNT_ID], [PARENT_ID], [NAME], [TYPE], [SALESREP],
   [SALESTEAM] FROM ACCOUNT;
   ```
7. Click **Preview** to view the first ten records returned by the query.

Using a parameter in the SQL
You can create a parameter to use in the Datasource field, then call that parameter as part of a query.

For example in the following SQL statement,

```
SELECT * FROM customer WHERE last_name = 'Smith'
```

you can replace the value of Smith with the parameter 'pLastName' so the SQL statement becomes:

```
SELECT * FROM customer WHERE last_name = '?pLastName?'
```

When creating a parameter, consider the following:

- You must initially create the TI process using an ODBC source. This will populate the the Variables tab. At that point, you can use the variable DATASOURCEQUERY to overwrite the query text box value in the Datasource tab.
- The number of columns from the returned set must match the number when the TI process was developed.
- The data type of columns must also match.
- It is important to enclose the parameter with single quotes when it is a string parameter. For a numeric parameter, do not use the single quotes, for example the query using a numeric could be

  ```
  SELECT
  * FROM customer WHERE last_name = ?pQuantity?
  ```

To create the parameter, use the Advanced Tab in the TurboIntegrator process dialog box to replace the default PO parameter with the parameter you want to use, for example: **pLastName**.

Identifying Variables
After querying the source data, you must identify the contents of each field in the query results.

Procedure

1. Click the **Variables** tab.

   Note that the Variable Name column has been filled with the correct column names from the database.
2. Change the selections in the Contents column to these selections.

Variable Name	**Contents**
ACCOUNT_ID	Ignore
PARENT_ID	Ignore
NAME	Element

Variable Name	Contents
TYPE	Consolidation
SALESREP	Consolidation
SALESTEAM	Consolidation

You are now ready to map variables.

Mapping Variables

Map variables by mapping elements to dimensions then mapping consolidation variables.

Procedure

1. Map elements to dimensions.
   a) Click the **Maps** tab, then click the **Dimensions** sub-tab.

      The single variable you identified as an element displays in the grid.
   b) In the Dimension column, type **DB.**
   c) Select **Create** from the Action drop-down menu.
   d) Select **Numeric** from the Element Type drop-down menu.
2. Map consolidation variables.
   a) Click the **Consolidations** sub-tab.

      TM1 correctly identifies each consolidation variable as mapping to the DB dimension.
   b) Set the Child Variable for each consolidation variable.

Cons. Variable	Child Variable
TYPE	SALESREP
SALESREP	NAME
SALESTEAM	TYPE

Saving and Executing the Process

To save and execute the process:

Procedure

1. Click the **Run** button ⚡.

   TM1 prompts you to save the process.
2. Save the process as create_DB_dimension.

   After a few seconds, you should see confirmation that the process executed successfully.
3. Close the TurboIntegrator window.
4. Open the Server Explorer.
5. Double-click the new **DB** dimension.

   The DB dimension opens in the Subset Editor.
6. Select **Edit**, **Sort**, **Hierarchy** from the Subset Editor menu bar to display the dimension elements and consolidations.

   The DB dimension contains over 40 elements and has four hierarchy levels.

Creating a Cube and Processing Data

The next example shows how to use TM1 TurboIntegrator to create a cube, dimensions, and elements and process data at the same time.

Defining the Data Source

Perform the following steps to define a data source.

Procedure

1. In the left pane of the Server Explorer, right-click the **Processes** icon and select **Create New Process**.

   The TurboIntegrator window opens.
2. Click the **Data Source** tab on the TurboIntegrator window.
3. Set the Data Source Type as **Text**; the Delimiter Type as **Delimited**; and the Delimiter as **Comma**.

   Ignore the Quote Char and Number of Title Records fields.
4. Make sure the Decimal Separator is period (.) and the Thousand Separator is comma (,).
5. Click the **Browse** button next to the Data Source Name field and select the file **newcube.csv** in your TI_data directory.
6. Click **Preview** to view the first ten records of the data source.

   Each record in newcube.csv contains 20 fields. You can scroll across the display grid to view all fields.

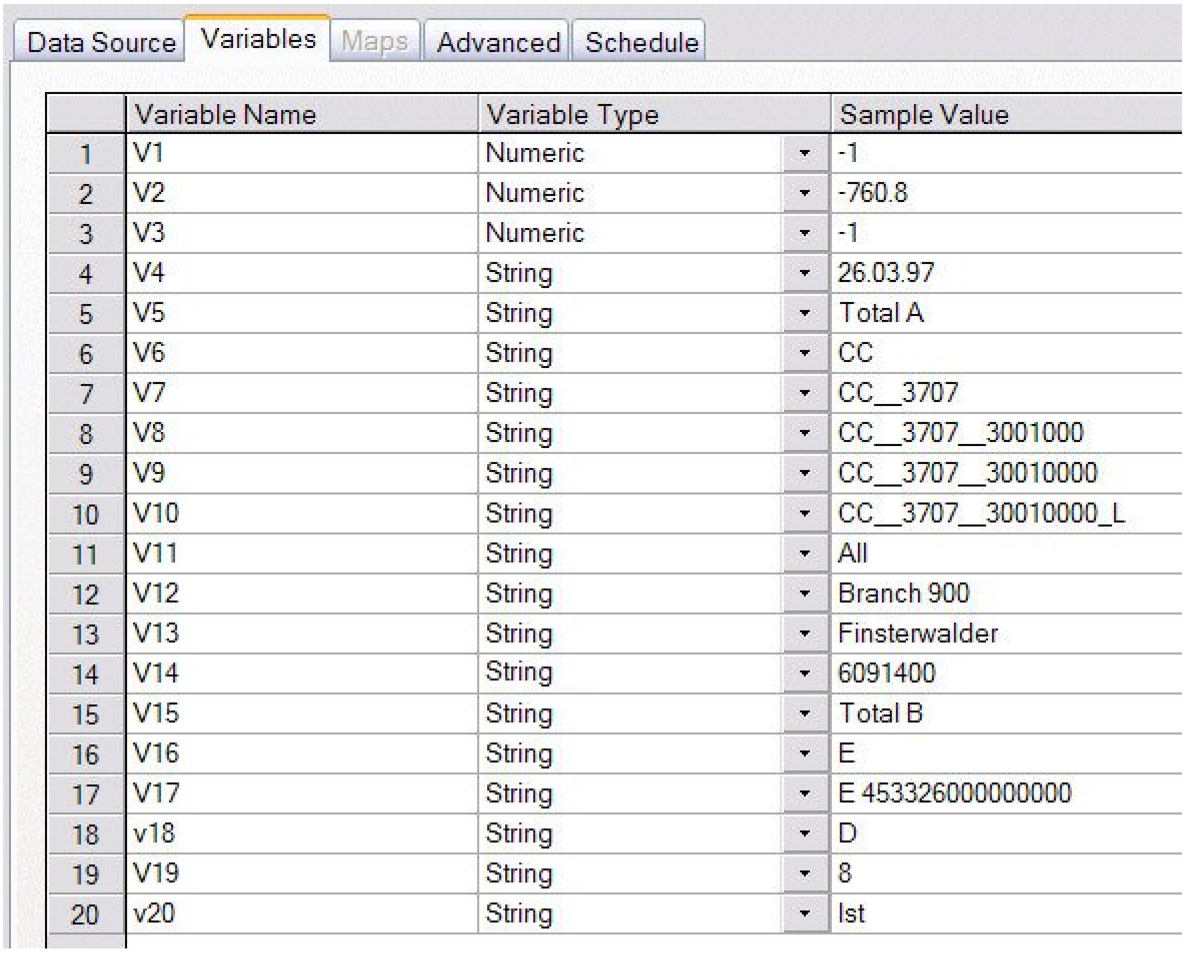
Identifying Variables

After loading the source data into TurboIntegrator, you must identify the contents of each field in the source.

Procedure

1. Click the **Variables** tab.

   Some variables will use the V*n* naming convention while others will use names corresponding to the first record in the source file.
2. To simplify the editing process, rename all variables using the V*n* convention. The first variable should be named V1, the second variable V2, and so on. When you are done, the Variables tab should appear as follows:



	Variable Name	Variable Type	Sample Value
1	V1	Numeric	-1
2	V2	Numeric	-760.8
3	V3	Numeric	-1
4	V4	String	26.03.97
5	V5	String	Total A
6	V6	String	CC
7	V7	String	CC__3707
8	V8	String	CC__3707__3001000
9	V9	String	CC__3707__30010000
10	V10	String	CC__3707__30010000_L
11	V11	String	All
12	V12	String	Branch 900
13	V13	String	Finsterwalder
14	V14	String	6091400
15	V15	String	Total B
16	V16	String	E
17	V17	String	E 453326000000000
18	v18	String	D
19	V19	String	8
20	v20	String	Ist

3. For each variable, select a type from the associated Variable Type drop-down list.

   For variables V1, V2, and V19, the type is **Numeric**. For all other variables, the type is **String**.

4. For each variable, select a content type from the associated Contents drop-down list. Refer to the following table to identify the content type for each variable.

Variable Name	Contents	Variable Name	Contents
V1	Data	V11	Consolidation
V2	Data	V12	Consolidation
V3	Data	V13	Consolidation
V4	Element	V14	Element
V5	Consolidation	V15	Consolidation
V6	Consolidation	V16	Consolidation
V7	Consolidation	V17	Element
V8	Consolidation	V18	Element
V9	Consolidation	V19	Element
V10	Element	V20	Element

Mapping Variables
You have identified variables for data, elements, and consolidations. Now you have to map the variables and provide instructions for creating a new cube.
Mapping the Cube
To provide cube mapping instructions:

Procedure

1. Click the **Maps** tab.
2. Click the **Cube** sub-tab.
3. Select **Create** for the Cube Action.
4. Type **NewCube** in the Cube Name field.
5. Select **Store Values** for the Data Action.
6. Do not turn on the Enable Cube Logging option.

   When you enable cube logging, TM1 logs changes to cube data during processing. You are creating a new cube, so there is no need to log changes.

Mapping Element Variables to Dimensions
You can now map all variables you identified as having an Element type to appropriate dimensions.

Procedure

1. Click the **Dimensions** sub-tab.
2. Using the following table as a guide, specify a Dimension, Action, and Element Type for each element variable.

Element Variable	**Dimension**	**Action**	**Element Type**
V4	date	Create	Numeric
V10	item	Create	Numeric
V14	customer	Create	Numeric
V17	job	Create	Numeric
V18	region	Create	Numeric
V19	agent	Create	Numeric
V20	book	Create	Numeric
Data variables	measure	Create	Numeric

   You can accept the default **Order in Cube** values for each variable.

Mapping Data Variables
You must now map variables you identified as having a Data type to individual elements.

Procedure

1. Click the **Data** sub-tab.
2. For data variable V1, enter **weight** as the element to which the variable will map.
3. For V2, enter **conversion**.
4. For V3, enter **pieces**.
5. In the Element Type column, select **Numeric** for all three elements.

Mapping Consolidation Variables
You must now map consolidation paths for all variables you identified as having a Consolidation content.

Procedure

1. Click the **Consolidations** sub-tab.
2. Using the following table as a guide, specify a Dimension and Child Variable for each consolidation variable.

Consolidation Variable	**Dimension**	**Child Variable**
V5	item	V6
V6	item	V7
V7	item	V8
V8	item	V9
V9	item	V10
V11	customer	V12
V12	customer	V13
V13	customer	V14
V15	job	V16
V16	job	V17

3. You can accept the default Weight and Component Order for all consolidation variables.

   You have now completed mapping to create new dimensions, insert elements and consolidations into the dimensions, create a new cube, and populate the cube with data.

Saving and Executing the Process
To save and execute the process:

Procedure

1. Click the **Run** button ⚡.

   TM1 prompts you to save the process.
2. Save the process as create_newcube.

   After a few seconds, you should see confirmation that the process executed successfully.
3. Open the Server Explorer and note that the cube NewCube has been created and populated, and that all required dimensions have been created.

   Browse the new cube (it is very sparsely populated) and examine the newly created dimensions.

Advanced Scripting

Use the **Advanced** tab of TurboIntegrator to create parameters that can be passed to a process at runtime or to edit process procedures, thereby enhancing the capabilities of TurboIntegrator. Procedures are edited by creating scripts that incorporate both TurboIntegrator functions and TM1 rules functions.

Editing the Prolog, Metadata, Data, and Epilog Procedures

You can enhance the capabilities of TurboIntegrator by editing the procedures that define the actions of a process. A procedure is a group of statements that manipulates TM1 data or metadata.

A process includes four procedures that are executed in sequence. Each procedure contains generated statements that are created based on the options you select elsewhere in the TurboIntegrator window. You can edit these procedures by adding your own statements that incorporate TurboIntegrator functions and Rules functions.

The procedures contained within a process are:

Tab	Description
Prolog	A series of actions to be executed before the data source is processed
Metadata	A series of actions that update or create cube, dimensions, and other metadata structures during processing.
Data	A series of data actions to be executed for each record in the data source.
Epilog	A series of actions to be executed after the data source is processed.

When editing procedures, keep in mind that each procedure is intended to execute certain types of actions at specific times in a process. Accordingly, you should create actions or statements that are appropriate for a given procedure.

For example, to export processed data to an ASCII file, you would add an ASCIIOutput function to the Data procedure. ASCIIOutput is a function that manipulates data, and it should be executed during processing. Therefore, the Data procedure is the correct location for the function.

Editing a Procedure

To edit a procedure:

Procedure

1. Click the **Advanced** tab on the TurboIntegrator window.
2. Click the sub-tab for the procedure you want to edit.
3. Enter your statements in the text box *before* the

   #****GENERATED STATEMENTS START**** line

   or *after* the

   #****GENERATED STATEMENTS FINISH**** line.

   You should not edit generated statements between these two lines.

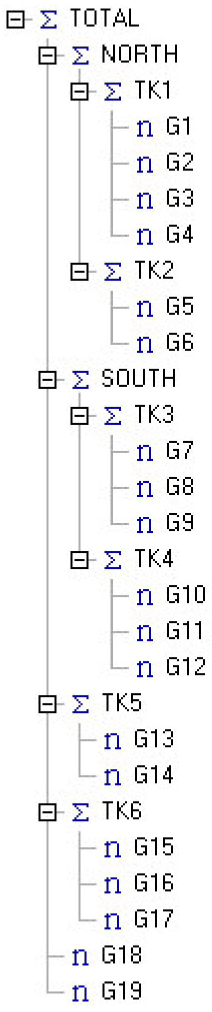
Creating a Dimension with Unbalanced Hierarchies

In this exercise, you will use the following input file to create a dimension with unbalanced hierarchies.

```
TOTAL,NORTH,TK1,G1
TOTAL,NORTH,TK1,G2
TOTAL,NORTH,TK1,G3
TOTAL,NORTH,TK1,G4
TOTAL,NORTH,TK2,G5
TOTAL,NORTH,TK2,G6
TOTAL,SOUTH,TK3,G7
TOTAL,SOUTH,TK3,G8
TOTAL,SOUTH,TK3,G9
TOTAL,SOUTH,TK4,G10
TOTAL,SOUTH,TK4,G11
TOTAL,SOUTH,TK4,G12
TOTAL,TK5,G13
TOTAL,TK5,G14
TOTAL,TK6,G15
```

```
TOTAL,TK6,G16
TOTAL,TK6,G17
TOTAL,G18
TOTAL,G19
```

The final result will look like this:



To begin creating the dimension:

Procedure

1. In the left pane of the Server Explorer, right-click the **Processes** icon and select **Create New Process**.

   The TurboIntegrator window opens.
2. Select the **Text** Data Source Type.
3. Click **Browse** next to the Data Source Name field and select **unbalanced.csv** in your TI_data directory.
4. Leave all other options on the Data Source tab at their default settings.
5. Click **Preview** to view the first ten records in the data source.

Identifying Variables

After loading the source data into TurboIntegrator, you must identify the contents of each field in the source.

Procedure

1. Click the **Variables** tab.
2. In the Contents column, select **Consolidation** for variables Total, North, and TK1.

3. Select **Element** for variable G1.

Mapping Variables
You have identified variables elements and consolidations. Now you have to map the variables to a dimension and define consolidation paths.

Procedure

1. Click the **Maps** tab.
2. Click the **Dimensions** sub-tab.
3. For element variable G1, enter **unbalanced** as the Dimension; **Create** for the Action; and **Numeric** for the Element Type.
4. Click the **Consolidations** sub-tab.
5. In the **Dimension** column, select **unbalanced** from the drop-down list for the three variables.
6. For the Cons. VariableTotal, select **North** as the Child Variable.
7. For the Cons. VariableNorth, select **TK1** as the Child Variable.
8. For the Cons. VariableTK1, select **G1** as the Child Variable.

Copying Generated Statements
TM1 generates statements dynamically as you change options in the TurboIntegrator window.

You are going to edit the generated statements on the Prolog and Metadata subtabs of the Advanced tab to accomodate an unblanced dimension hierarchy. To make things a little easier, you will copy and paste the generated statements so that they will be available after you change options in the TurboIntegrator window.

Procedure

1. Click the **Advanced** tab, then the **Prolog** sub-tab.
2. Copy the DimensionDestroyand DimensionCreate functions from between the comment lines

   ```
   #****GENERATED STATEMENTS START****
   ```

   ```
   #****GENERATED STATEMENTS FINISH****
   ```

   and paste them below the comment lines.

   ```
   #****GENERATED STATEMENTS START****
   ```

   ```
   DIMENSIONDESTROY('unbalanced');
   ```

   ```
   DIMENSIONCREATE('unbalanced');
   ```

   ```
   DIMENSIONSORTORDER('unbalanced','ByInput','ASCENDING','ByInput','ASCENDING');
   ```

   ```
   ****GENERATED STATEMENTS FINISH****
   ```

   ```

   ```

   ```
   DIMENSIONDESTROY('unbalanced');
   ```

   ```
   DIMENSIONCREATE('unbalanced');
   ```
3. Click the **Metadata** sub-tab.

   There are two functions:

The DimensionElementInsert function adds a simple (leaf) element to a dimension. You can use this function to add both numeric and string elements.

The DimensionElementComponentAdd function adds a component (child) to a consolidated element.

4. Copy all the generated statements and paste them below the last comment line.

```
#****GENERATED STATEMENTS START****

DIMENSIONELEMENTINSERT('unbalanced',",G1,'n');

DIMENSIONELEMENTINSERT('unbalanced',",TOTAL,'c');

DIMENSIONELEMENTINSERT('unbalanced',",NORTH,'c');

DIMENSIONELEMENTINSERT('unbalanced',",TK1,'c');

DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);

DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);

DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);

****GENERATED STATEMENTS FINISH****


DIMENSIONELEMENTINSERT('unbalanced',",G1,'n');

DIMENSIONELEMENTINSERT('unbalanced',",TOTAL,'c');

DIMENSIONELEMENTINSERT('unbalanced',",NORTH,'c');

DIMENSIONELEMENTINSERT('unbalanced',",TK1,'c');

DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);

DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);

DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);
```

Removing the Generated Statements Permanently
To permanently remove generated statements:

Procedure

1. Click the **Variables** tab and change the selections in the Contents column to **Other**.

   When a variable is identified as Other, the variable is available to be used in advanced scripts. If a variable is identified as Ignore, it is not processed by TurboIntegrator and thus cannot be referenced in advanced scripts.
2. To verify that the statements have been removed, click on the **Advanced** tab, then the **Prolog** and **Metadata** sub-tabs.

The statements should appear as follows:

```
Prolog>
```

```
#****GENERATED STATEMENTS START****
```

```
#****GENERATED STATEMENTS FINISH****
```

```
DIMENSIONDESTROY('unbalanced');
```

```
DIMENSIONCREATE('unbalanced');
```

```
Metadata>
```

```
#****GENERATED STATEMENTS START****
```

```
#****GENERATED STATEMENTS FINISH****
```

```
DIMENSIONELEMENTINSERT('unbalanced','',G1,'n');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',TK1,'c');
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);
```

Editing the TurboIntegrator Statements
Examine the script that currently exists on the Metadata sub-tab, which appear as follows.

```
DIMENSIONELEMENTINSERT('unbalanced','',G1,'n');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',TK1,'c');
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);
```

This script, which was generated based on the first record in unbalanced.csv, is valid for records that contain four fields. The script creates dimension elements from each field in the source, then creates a hierarchy. The script, however, is not valid for records containing less than four fields.

Because the source file unbalanced.csv contains records of varying length, you must modify the script to evaluate each record in the source. The script should determine the correct level of consolidation and specify an appropriate consolidation path for each possible level of consolidation. This can be accomplished by editing the script to include an IF function, which allows you to execute other TurboIntegrator statements based on defined conditions.

Procedure

1. Click the **Advanced** tab, then the **Metadata** sub-tab.
2. Insert the line

   ```
   IF (G1@<>'');
   ```

   before the first DIMENSIONELEMENTINSERT statement. This IF statement indicates that if the string variable G1*is not* empty, the statements that follow should be executed. If V4*is* empty, processing should skip to the next conditional statement.

   The Metadata sub-tab now appear as follows.

   ```
   #****GENERATED STATEMENTS START****
   ```

   ```
   #****GENERATED STATEMENTS FINISH****
   ```

   ```
   IF (G1@<>'');
   ```

   ```
   DIMENSIONELEMENTINSERT('unbalanced','',G1,'n');
   ```

   ```
   DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
   ```

   ```
   DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
   ```

   ```
   DIMENSIONELEMENTINSERT('unbalanced','',TK1,'c');
   ```

   ```
   DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
   ```

   ```
   DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
   ```

   ```
   DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);
   ```

   When IF (G1@<>'') is true, TurboIntegrator inserts three consolidated elements (Total, North, TK1) and a single numeric element (G1) into the unbalanced dimension. TurboIntegrator also creates a four-level hierarchy where Total is the parent of North, North is the parent of TK1, and TK1 is the parent of G1.
3. Insert the line

   ```
   ELSEIF (TK1@<>'');
   ```

   after the last DIMENSIONELEMENTCOMPONENTADD statement.

   This conditional ELSEIF, statement indicates that if the string variable V3*is not* empty, the statements that follow should be executed. If V3 *is* empty, processing should skip to the next conditional statement.
4. You must now insert statements to be executed when ELSEIF (TK1@<>'') is true.

When ELSEIF (TK1@<>'') is true, the source record contains three fields. Accordingly, the statements should create a dimension element from each field, then create a hierarchy of three levels.

5. Insert the following statements immediately after ELSEIF (TK1@<>'');.

```
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',TK1,'n');
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
```

When IF (TK1@<>'') is true, TurboIntegrator inserts two consolidated elements (TOTAL, NORTH) and a single numeric element (TK1) into the unbalanced dimension. TurboIntegrator also creates a three-level hierarchy where TOTAL is the parent of NORTH and NORTH is the parent of TK1.

6. Insert the line

```
ELSE;
```

after the last DIMENSIONELEMENTCOMPONENTADD statement.

7. You must now insert statements to be executed when processing reaches the ELSE statement. (This occurs when both IF (G1@<>'') and ELSEIF (TK1@<>'') are false.)

When processing reaches the ELSE statement, the source record contains two fields. The statements you insert should create a dimension element from each field, then create a hierarchy of two levels.

8. Insert the following statements immediately after ELSE;.

```
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
```

```
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'n');
```

```
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
```

These statements instruct TurboIntegrator to insert the consolidated element TOTAL and the numeric element NORTH into the unbalanced dimension, and to create a hierarchy where TOTAL is the parent of NORTH.

9. Insert the line

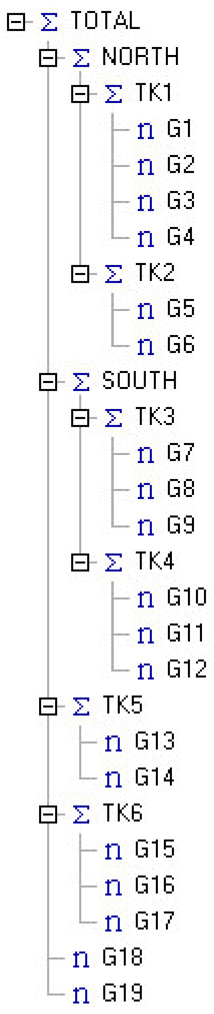
```
ENDIF;
```

after the final DIMENSIONELEMENTCOMPONENTADD statement. ENDIF indicates the end of the IF statement.

When you are done, the completed Metadata sub-tab should appear as follows:

```
#****GENERATED STATEMENTS START****
#****GENERATED STATEMENTS FINISH****
IF (G1@<>'');
DIMENSIONELEMENTINSERT('unbalanced','',G1,'n');
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
DIMENSIONELEMENTINSERT('unbalanced','',TK1,'c');
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TK1,G1,1.000000);
ELSEIF (TK1@<>'');
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'c');
DIMENSIONELEMENTINSERT('unbalanced','',TK1,'n');
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
DIMENSIONELEMENTCOMPONENTADD('unbalanced',NORTH,TK1,1.000000);
ELSE;
DIMENSIONELEMENTINSERT('unbalanced','',TOTAL,'c');
DIMENSIONELEMENTINSERT('unbalanced','',NORTH,'n');
DIMENSIONELEMENTCOMPONENTADD('unbalanced',TOTAL,NORTH,1.000000);
ENDIF;
```

10. Choose **File**, **Save** and name the process create_unbalanced_dim.
11. Choose **File**, **Run** to execute the process.
12. To verify that the dimension was built correctly, open the unbalanced dimension in the Dimension Editor. It should look like the following image.



Creating Subsets

In this exercise you create subsets for the dimension newdim, which is created by the dimension process.

Procedure

1. Open the process **subsets** in the TurboIntegrator window.

   You might have to edit the data source to point to region.csv in your TI_data directory. If you change the data source, you will be prompted to specify how process variables should be handled. Select **Keep All Variables**.

   This example uses the TM1 TurboIntegrator functions SubsetCreate() and SubsetElementInsert() to create and populate dimension subsets.

   The preview of the source file looks like this:

V0	**V1**	**V2**	**V3**	**V4**
Sweden	Scandanavia	Europe	International	Europe
Norway	Scandanavia	Europe	International	Europe
Denmark	Scandanavia	Europe	International	Europe
France	Europe	International	Worldwide	Europe
Germany	Europe	International	Worldwide	Europe

V0	**V1**	**V2**	**V3**	**V4**
UK	Europe	International	Worldwide	Europe
Ireland	Europe	International	Worldwide	Europe
Holland	Europe	International	Worldwide	Europe
Spain	Europe	International	Worldwide	Europe
Italy	Europe	International	Worldwide	Europe

Here are the scripts for the process subsets:

```
Prolog>
```

```
#****GENERATED STATEMENTS START****
```

```
#****GENERATED STATEMENTS FINISH****
```

```
SubsetCreate('NewDim','Europe');
```

```
SubsetCreate('NewDim','US');
```

```
SubsetCreate('NewDim','ROW');
```

```
Metadata>
```

```
#****GENERATED STATEMENTS START****
```

```
#****GENERATED STATEMENTS FINISH****
```

```
SubsetElementInsert('NewDim',V4,V0,0);
```

2. Execute the process.
3. In the Server Explorer, expand the newdim dimension and view the newly created subsets.

Creating Attributes

The AttrPutS function assigns a value to a string element attribute. If you want to assign the string Europe to the Continent attribute of the region Sweden in the NewDim dimension, you would write the AttrPutS function this way:

```
AttrPutS('Europe','NewDim','Sweden','Continent');
```

Procedure

1. Open the **Attributes** process in TurboIntegrator.

   You might have to edit the data source to point to region.csv in your TI_data directory. If you change the data source, you will be prompted to specify how process variables should be handled. Select **Keep All Variables**.
2. Click the **Variables** tab.

   Note that V4 and V5 have been identified as Attribute.

3. Click the **Formula** cell for V5.

   It reads V5=V0|V4;

   This formula concatenates the values of the V4 and V5 variables.
4. Click the **Maps** tab and the **Attributes** sub-tab.

   The attribute type for variable V4 has been defined as Text and the type for V5 as Alias.
5. Click the **Advanced** tab and the **Data** sub-tab to show the generated statements and two additional statements.

   ```
   #****GENERATED STATEMENTS START****
   ```

   ```
   V5=v0|v4;
   ```

   ```
   AttrPutS(V4,'newdim',V0,'continent');
   ```

   ```
   AttrPutS(V5,'newdim',V0,'cont');
   ```

   ```
    #****GENERATED STATEMENTS FINISH****
   ```

   ```
   AttrPutS(V4,'newdim',V1,'continent');
   ```

   ```
   AttrPutS(V4,'newdim',V2,'continent');
   ```

   The two statements above were added manually because V1 and V2 were not declared as contents in the Variables tab. They must, however, be assigned the text attribute Continent.
6. Save and execute the Attributes process.

Viewing the Attributes
After assigning an attribute value, you can view the assignment as follows.

Procedure

1. In the Server Explorer, double-click the **newdim** dimension to open the Subset Editor.
2. Click **Subset All** .
3. Choose **Edit**, **Filter By**, **Attribute** from the menu to display the Filter by Attribute dialog box.
4. Select an attribute value from the drop-down list on the Filter by Attribute dialog box to display all regions for a specific continent in the Subset Editor.

Appendix B. TurboIntegrator Reserved Words

This appendix lists IBM Cognos TM1 TurboIntegrator reserved words. To prevent errors in your TurboIntegrator scripts, you should avoid creating variables with names that match any of the words listed in the following tables.

There are four categories of reserved words in TurboIntegrator:

- Rule Function Names
- Process Function Names
- Implicit Variable Names
- TurboIntegrator Keywords

Rule function names

These are the reserved words for TM1 rule functions:

- ABS
- ACOS
- ASIN
- ATAN
- ATTRN
- ATTRS
- AVG
- BANNR
- BDATE
- BDAYN
- CAPIT
- CENTR
- CHAR
- CNT
- CODE
- COL
- Consolidate Children
- COS
- DATE
- DATES
- DATFM
- DAY
- DAYNO
- DBG16
- DBGEN
- DELET
- DFRST
- DIMIX
- DIMNM
- DIMSIZ
- DISPLY
- DNEXT
- DNLEV
- DTYPE
- DYS

- ELCOMP
- ELCOMPN
- ELISANC
- ELISCOMP
- ELISPAR
- ELLEV
- ELPAR
- ELPARN
- ELWEIGHT
- EXP
- FILL
- FV
- HEX
- IF
- INSRT
- INT
- IRR
- ISLEAF
- ISUND
- LIN
- LN
- LOG
- LONG
- LOOK
- LOWER
- MAX
- MEM
- MIN
- MOD
- MONTH
- MOS
- NCELL
- NOW
- NPV
- PAYMT
- PV
- RAND
- RIGHT
- ROUND
- ROUNDP
- SCAN
- SCELL
- SIGN
- SIN
- SLEEP
- SQRT
- STDDV
- STR
- SUBSIZ
- SUBST
- SUM
- TABDIM

- TAN
- TIME
- TIMST
- TIMVL
- TODAY
- TRIM
- UNDEF
- UPPER
- VAR
- WHOAMI
- WIDTH
- YEAR
- YRS

Process function names

These are the TurboIntegrator process function names:

- AddClient
- AddGroup
- AllowExternalRequests
- ASCIIDelete
- ASCIIOutput
- AssignClientPassword
- AssignClientToGroup
- AttrDelete
- AttrInsert
- AttrPutN
- AttrPutS
- AttrToAlias
- BatchUpdateFinish
- BatchUpdateStart
- CellGetN
- CellGetS
- CellIsUpdateable
- CellPutN
- CellPutProportionalSpread
- CellPutS
- ChoreQuit
- CubeCreate
- CubeDestroy
- CubeExists
- CubeGetLogChanges
- CubeLockOverride
- CubeProcessFeeders
- CubeSetConnParams
- CubeSetIsVirtual
- CubeSetLogChanges
- CubeSetSAPVariablesClause
- CubeSetSlicerMembers
- CubeUnload
- DeleteClient

- DeleteGroup
- DimensionCreate
- DimensionDeleteAllElements
- DimensionDestroy
- DimensionEditingAliasSet
- DimensionElementComponentAdd
- DimensionElementComponentDelete
- DimensionElementDelete
- DimensionElementInsert
- DimensionElementInsertByAlias
- DimensionElementPrincipalName
- DimensionExists
- DimensionSortOrder
- ElementSecurityGet
- ElementSecurityPut
- EncodePassword
- ExecuteCommand
- ExecuteProcess
- Expand
- FileExists
- GetProcessErrorFileDirectory
- GetProcessErrorFilename
- IsNull
- ItemReject
- ItemSkip
- LockOff
- LockOn
- LogOutput
- NumberToString
- NumberToStringEx
- NumericGlobalVariable
- NumericSessionVariable
- ODBCClose
- ODBCOpen
- ODBCOutput
- ProcessBreak
- ProcessError
- ProcessExitByBreak
- ProcessExitByChoreQuit
- ProcessExitByQuit
- ProcessExitMinorError
- ProcessExitNormal
- ProcessExitOnInit
- ProcessExitSeriousError
- ProcessExitWithMessage
- ProcessQuit
- PublishView
- RemoveClientFromGroup
- ReturnSQLTableHandle
- ReturnViewHandle
- RuleLoadFromFile
- SaveDataAll

- SecurityRefresh
- ServerShutDown
- SetChoreVerboseMessages
- StringGlobalVariable
- StringSessionVariable
- StringToNumber
- StringToNumberEx
- SubsetAliasSet
- SubsetCreate
- SubsetCreateByMDX
- SubsetDeleteAllElements
- SubsetDestroy
- SubsetElementDelete
- SubsetElementInsert
- SubsetExists
- SubsetFormatStyleSet
- SubsetGetElementName
- SubsetGetSize
- SubsetIsAllSet
- SubsetMDXGet
- SubsetMDXSet
- SwapAliasWithPrincipalName
- ViewColumnDimensionSet
- ViewColumnSuppressZeroesSet
- ViewConstruct
- ViewCreate
- ViewDestroy
- ViewExists
- ViewExtractSkipCalcsSet
- ViewExtractSkipConsolidatedStringsSet
- ViewExtractSkipRuleValuesSet
- ViewExtractSkipZeroesSet
- ViewRowDimensionSet
- ViewRowSuppressZeroesSet
- ViewSetSkipCalcs
- ViewSetSkipRuleValues
- ViewSetSkipZeroes
- ViewSubsetAssign
- ViewSuppressZeroesSet
- ViewTitleDimensionSet
- ViewTitleElementSet
- ViewZeroOut
- WildcardFileSearch

Implicit variable names

These are the implicit variable names for TurboIntegrator:

- DatasourceASCIIDecimalSeparator
- DatasourceASCIIDelimiter
- DatasourceASCIIHeaderRecords
- DatasourceASCIIQuoteCharacter

- DatasourceASCIIThousandSeparator
- DatasourceCubeview
- DatasourceDimensionSubset
- DatasourceNameForClient
- DatasourceNameForServer
- DatasourceODBOCatalog
- DatasourceODBOConnectionString
- DatasourceODBOCubeName
- DatasourceODBOHierarchyName
- DatasourceODBOLocation
- DatasourceODBOProvider
- DatasourceODBOSAPClientId
- DatasourceODBOSAPClientLanguage
- DatasourcePassword
- DatasourceQuery
- DatasourceType
- DatasourceUseCallerProcessConnection
- DatasourceUsername
- MinorErrorLogMax
- NValue
- OnMinorErrorDoItemSkip
- SValue
- Value_Is_String

TurboIntegrator keywords

These are the reserved TurboIntegrator keywords:

- break
- else
- elseif
- end
- endif
- if
- while

Notices

©

Product Information

This document applies to IBM Planning Analytics 2.0.0 and may also apply to subsequent releases.

Copyright

Index