Bedrock

Table of contents

Documentation and Downloads

Documentation and Downloads contains all the help documentation associated with the Bedrock TI processes and Best Practice TM1 whitepapers.

Click on the content tree to the left for links.

Please email info@bedrocktm1.org for help or feedback.

Quick Downloads

- Bedrock Turbo Integrator
  - White Paper - The Modular Approach
  - Latest Bedrock TM1 TI release - zip file
- Best Practice
  - White Paper - Best Practice Cube Design
  - White Paper - Best Practice Rules
  - White Paper - Best Practice Turbo Integrator
- Project Management
  - White Paper - Managing TM1 Projects

Release 2.0

This Release

In this version, we aim to make Bedrock more “TM1 version” and “system-type” agnostic. Object locking is dependent on both your version of TM1 and how you use it, i.e. your propensity for creating and destroying objects, how concurrent they are and whether they are system or user-driven. This brings into play trade-offs of performance and system maintenance which the TM1 developer can flexibly invoke via Bedrock's multi-switch parameters.

Thus we drive towards giving as much flexibility as possible to the developer; in this release this new functionality chiefly is manifested in a new Bedrock “core” process called Bedrock.Cube.View.Create that is used by the clearing, copying and exporting “calling” processes.

Instead of using Bedrock named views and subsets the TM1 developer can nominate the name of the view to be used by the Bedrock process. If no view or subset name is provided then a Bedrock name will be used with a date stamp and random number. This is to ensure that there is no concurrency issue with two users using the same object at the same time. There is also an option for the TM1 developer to decide if the Bedrock view and/or subset are to be cleaned up at the end of the Bedrock process.

This improvement has been made to all Bedrock processes in Bedrock 2.0.

Numerous fixes have been made to existing Bedrock processes. View http://code.bedrocktm1.org/ for version control between Bedrock 1.0 and 2.0.

How the processes now work

Two new processes have been created for clearing and exporting data, these are called **Bedrock.Cube.Data.Clear** and **Bedrock.Cube.Data.Export**, these supersede **Bedrock.Cube.Data.ZeroOut** and **Bedrock.Cube.Data.ExportToFile**. Each of these processes in addition to **Bedrock.Cube.Data.Copy** now have a new parameter called **pDestroyTempObj**, this parameter determines whether the views and subsets are deleted after use:

- 0 = Retain View and Subsets
- 1 = Delete View and Subsets
- 2 = Delete View only

In each of these 3 processes you can pass in the name of the view so you can reuse it appropriately if you want to take advantage of Parallel Interaction (9.5.2+). For example you need to use a view name that won't be used by another process at the same time, i.e. *} GL.Hourly.Load*

New Processes

BEDROCK.CUBE.VIEW.CREATE

This process is now one of the foundation processes for a number of other Bedrock processes. It takes the concept of a filter first introduced in **Bedrock.Cube.Data.ZeroOut** and makes it easier to read and reusable for other processes. The process takes a string filter and then does the heavy lifting for you creating a view that can be used for multiple purposes. Below are some examples of filters from the Bedrock documentation:

- **Year: 2010** - Year 2010 only
- **Year: 2010 & Version: Actual** - Year 2010 and Version Actual
- **Year: 2010 + 2011** - Year 2010 and Year 2011
- **Year: 2010 + 2011 & Month: Jan + Feb** - Years 2010 and 2011 for the Months of January and February
- **Year: 2010 & Month: All Months** - If the pSkipConsols parameter is set to 1, any consolidated elements passed to the filter will be converted to their N level children. In this case it will be Year 2010 and January, February, March, April, May, etc.

  Hot Tip: When building a Filter String, use the debug functionality for a char by char feedback to flat file.

BEDROCK.CUBE.DATA.CLEAR

Replaces Bedrock.Cube.Data.ZeroOut. Uses the new filter concept by calling Bedrock.Cube.View.Create

BEDROCK.CUBE.DATA.EXPORT

Replaces **Bedrock.Cube.Data.Export.** Now uses the filter concept by calling Bedrock.Cube.View.Create

BEDROCK.CUBE.VIEWANDSUBSETS.DELETE

Is used in conjunction with **Bedrock.Cube.View.Create** to delete the views and subsets using a common name. Use this process in your Epilog to clean up a view that was cleared in the Prolog tab.

BEDROCK.DIM.CLONEFROMSUBSET

Creates a new dimension based on a subset from another dimension.

BEDROCK.DIM.CLONEFROMSUBSET.FLAT

Creates a new dimension with a flat hierarchy based on a subset in another dimension.

BEDROCK.SERVER.DATADIR.BACKUP

Makes a copy of all objects from the data directory to the specified backup directory.

BEDROCK.SERVER.DATADIR.LISTCONTENTS

Outputs to text file the contents of the data directory - very helpful for documentation.

Bedrock TM1 Turbo Integrator

Welcome to Bedrock TM1 Turbo Integrator. Bedrock TM1 TI is a collection of Turbo Integrator (TI) processes that are designed to make building models easier and faster than ever before.

Each TI is a fully contained, auditable, multi-function code block with parameter switches permitting the user to perform numerous tasks. Well over 10,000 lines of code are contained within these processes meaning you don't have to write them - just use them at the right time for the right job.

Modular Turbo Integrator Coding

The Bedrock TM1 TI supports the modular coding approach to building TI processes. Common functions are encapsulated in parameterised TI processes. Any custom TI process can call particular Bedrock TM1 TI processes to perform required functions.

A common example is clearing an area of the cube before loading data from an external source. Before the data is loaded into the cube, a portion of the cube must be cleared of values. Traditionally, all code that is required to clear that portion of the cube is written in a custom TI process. By using Bedrock TM1 TI, the process Bedrock.Cube.Data.ZeroOut can be called from within any custom TI process. Using the required parameters to restrict the specific area of cube, the process will build the required view, clear the data, and when complete, continue on to the custom code.

Changes in the requirements are easy to manage using Bedrock TM1 TI. In the above example a TM1 developer can quickly change the parameters to clear more or less data in the cube. The traditional approach would require the TM1 developer to review all the code in the custom process, add the required code and then re-test the results. By using Bedrock TM1 TI, the entire process is made simple and easy to manage.

Best Practice

All Bedrock TM1 TI processes that have been developed will execute in the most efficient manner according to TM1 Black Belt techniques such as server locking. The modular approach to coding can produce time saving results that cannot be achieved using traditional TI coding. For example, the process Bedrock.Dim.Sub.Create will build subsets in the 'metadata' tab instead of the 'prolog' tab. The advantage being that the components of the subset are held in memory until the very end, at which point, the subset is written to disk. This is more efficient than the traditional manner (in the prolog using a 'While...End' loop) in which the subset file is saved to disk on each change, causing resource inefficiencies.

Bedrock TM1 TI may require changes in the future to reflect changes in best practice. This could be due to a new release of TM1 or an increased understanding of existing implementations. By using Bedrock TM1 TI, the impact of changes is minimal as it is easier to change a single Bedrock TM1 TI, than it is to review, change and test many instances of custom code to achieve the same outcome.

More than just code

Bedrock TM1 TI contains more than just processes to support modular coding. It includes a number of utilities that will help with performing everyday tasks. For example, in situations which require different security for the same dimension, the process Bedrock.Dim.Clone will clone the dimension to enable changes to be made to the security definition, and the process Bedrock.Cube.Dimension.Replace will replace the dimension in the cube.

What the future holds

Bedrock TM1 TI is for anyone serious about getting the best out of TM1 in a simple and ordered environment.

At BedrockTM1.org we are committed to ongoing development and invite everyone to use and test the TI functions. We also ask that those that benefit make suggestions and contribute to improving Bedrock TM1 TI as the entire TM1 community will benefit from our work together.

Getting Started

Getting Started

Follow these simple steps.

1. Download the Bedrock TM1 TI zip file on the homepage at BedrockTM1.org or pick up the latest individual process files in the Source Control System in the Share page of BedrockTM1.org.
2. Unzip or copy the files to your TM1 data directory.
3. Restart your TM1 server.
4. Refer to the online help to use the processes on your TM1 server.

System requirements

System requirements

All Bedrock TM1 TIs have been extensively tested on multiple TM1 versions and environments. Nevertheless it is good and common practice to test independently before being implemented in any production environment TM1 model.

Getting help

Getting help

Please email info@bedrocktm1.org for help or feedback.

Error trapping example

Error trapping example

Bedrock processes have their own auditing yet you may want to trap errors externally. Below is an example of using a BedrockTM1 TI process and trapping its return state;

```
NumericGlobalVariable('nRet');

# Data ZeroOut

 sProc = 'Bedrock.Cube.Data.ZeroOut';

 nRet = ExecuteProcess ( sProc , 'pCube' , sCube ,'pFilter' , 'Version+Actual&Year+2009+2008' , 'pDebug' , pDebug );

 #Region "Block Audit"

 IF( nRet = ProcessExitNormal() );

   sRet = 'Process Completed Successfully';

 ELSEIF( nRet = ProcessExitByChoreQuit() );

   sRet = 'Exit by Chore Quit';

 ELSEIF( nRet = ProcessExitMinorError() );

   sRet = 'Exit with Minor Error';

 ELSEIF( nRet = ProcessExitByQuit() );

   sRet = 'Exit by Quit';

 ELSEIF( nRet = ProcessExitWithMessage() );

   sRet = 'Exit with Message';

 ELSEIF( nRet = ProcessExitSeriousError() );

   sRet = 'Exit with Serious Error';

 ELSEIF( nRet = ProcessExitOnInit() );

   sRet = 'Exit on Initiation';

 ELSEIF( nRet = ProcessExitByBreak() );

   sRet = 'Exit by Break';

 Endif;

 CellPutS( sRet ,'}Docu_Processes' , sProc , 'Process State' );

#End Region
```

Debugging

All of the Bedrock TM1 TI processes include debugging capabilities. This gives the developer more information than traditionally found in the TM1 logs in the event of an error (or other unexpected behaviour) when executing TI processes. This information will assist in identifying the problem source and thus help with troubleshooting errors.

To enable the debugging capability when executing a process, a value must be specified for the pDebug parameter. There are three possible values for this parameter (0,1& 2):-

0 - Run the process normally with no debugging. This is the default value.

1 - Run the process normally and write information out to the debug files.

2 - Run the process and write information out to the debug files but don't perform any updates.

When debugging is enabled (option 1 or 2), the relevant debug output will be sent to debug files in the logging directory of the TM1 server.

The debug file name defaults to '[processname].[timestamp].[tab].debug', for example 'Bedrock.Cube.Clone.30-03-2011 12:00:00.Prolog.Debug'.

There is a separate file for each tab in the process: Prolog, Metadata, Data, and Epilog.

The content of the debug files will vary from process to process. There are some standard outputs for all processes:-

- Process start time
- Process finish time
- Parameter values

Other debugging information may include the variable values, parameter errors and customised debug messaging.

What's new

This is the initial release of Bedrock TM1 TI and includes processes built in the following classes:

- Cubes
- Dimensions
- Subsets
- Server Administration

Cubes

Cubes

The Bedrock.Cube processes are used for tasks such as copying and clearing cubes, importing and exporting cube data, replacing dimensions and working with views.

Bedrock.Cube.Clone

Bedrock.Cube.Clone

This Bedrock TM1 TI will clone an existing cube. The dimensional structure of the cube is copied to the clone cube. Data and rules may also be copied to the clone cube.

Limited to a cube with a maximum of 27 dimensions.

Parameters

pSourceCube	String		The original cube name that is to be cloned.
pTargetCube	String		The name of the clone cube. If this parameter is blank the default clone cube name is the source cube name suffixed with '_Clone'.
pIncludeRules	Boolean	1	Clone the rules from the original cube to the clone cube. The clone process does not change the rules from the original cube. The clone rules should be reviewed to ensure cube references are correct after cloning.
pIncludeData	Boolean	0	Clone the data from the source cube.
pSourceView	String		The name of an existing view in the source cube. This view will be used to restrict the amount of data copied from the original cube. If this parameter is *null* then all the data in the cube will be copied.
pRuleValues	Boolean	1	Use 1 to exclude rule calculated data. Use 0 to include rule calculated data. This value is set to 1 if the original cube rules are cloned as per the pIncludeRules parameter.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Clone',
    'pSourceCube','General Ledger',
    'pTargetCube','General Ledger Cube Clone',
    'pIncludeRules',1,
    'pIncludeData',1,
    'pSourceView','2011 Actual Data',
    'pRuleValues',1,
    'pDebug',0
    );
```

Bedrock.Cube.Data.Clear

Bedrock.Cube.Data.Clear

This Bedrock TM1 TI will build a temporary view for a single cube that is to be zeroed out. The temporary view can be restricted by nominating one or more dimensions and elements. Using the filter parameter, nominate the dimension followed by the elements to be included in the temporary subset separated by the Element Delimiter (pElementDelim). If more than one dimension is required, then separate each dimension and element set using the Dimension

Delimiter (pDimensionDelim).

The filter can be based on multiple dimensions and multiple views, here are some standard examples using the standard delimiters:

- **Year: 2010** - Year 2010 only
- **Year: 2010 & Version: Actual** - Year 2010 and version Actual
- **Year: 2010 + 2011** - Year 2010 and Year 2011
- **Year: 2010 + 2011 & Month: Jan + Feb** - Years 2010 and 2011 for the Months of January and February
- **Year: 2010 & Month: All Months** - If the pSkipConsols parameter is set to 1, any consolidated elements passed to the filter will be converted to their N level children. In this case it will be Year 2010 and January, February, March, April, May, etc.

Filter Tips

- Principal or Alias's can be used for elements
- Spaces are ignored
- Consolidated elements are converted to the N level children when pSkipConsols is set to 1 (the default).
- You can use multiple characters for the delimiters, this is important for dimensions were special characters are used. If your dimension has special characters it is suggested that you use multiple character delimiters, i.e.:

  Year:: 2010 ++ 2011 && Version:: Actual

  OR

  Year:= 2010 ++ 2011 && Version:= Actual

  It is rare for multiple special characters to be side by side.

If there is no filter parameter provided, the entire cube will be cleared.

Parameters

pCube	String		The cube name where the view is to be created.
pView	String		The name to use for the temporary view subsets created by the process. If omitted or blank, a view name consisting of a time stamp and random number is used.
pFilter	String		Restrict the portion of the cube to be cleared by entering

			dimension and element sets. **For example:** Version : Actual & Year : 2011 & Month + Sep + 2nd QTR
pDimensionDelim	String	&	The delimiter between dimensions.
pElementStartDelim	String	:	The delimiter at the end of the dimension name and the start of the list of elements.
pElementDelim	String	+	The delimiter between elements.
pDeleteTempObj	Boolean	1	Use 0 to retain temporary views and subsets created by the process. Use 1 to delete temporary views and subsets created by the process. Use 2 to delete only the temporary views created by the process.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.Clear',
    'pCube','General Ledger',
    'pView','',
    'pFilter','Year: 2011 + 2012 & Version: Budget',
    'pDimensionDelim','&',
    'pElementStartDelim',':',
    'pElementDelim','+',
    'pDeleteTempObj',1,
    'pDebug',0
    );
```

Bedrock.Cube.Data.Copy

Bedrock.Cube.Data.Copy

This Bedrock TM1 TI will copy data within a cube from one element in a dimension to another element in the same dimension.

Limited to a cube with a maximum of 27 dimensions.

Parameters

pCube	String		The name of the cube where the data exists.
pViewSource	String		The name to use for the temporary source view and subsets created by the process. If omitted or blank, a view name consisting of a time stamp and random number is used.
pViewTarget	String		The name to use for the temporary target view and

			subsets created by the process. If omitted or blank, a view name consisting of a time stamp and random number is used.
pDimension	String		The dimension where the source and target elements exist.
pSourceElement	String		The element in the dimension from which the data is copied.
pTargetElement	String		The element in the dimension where the data is to be copied to.
pSkipRules	Boolean	1	Use 0 to include rule calculated data in the copied data. Use 1 to exclude rule calculated data from the copied data..
pZeroTarget	Boolean	1	Use 1 to zero out data in the target element before copying the data.
pZeroSource	Boolean	0	Use 1 to zero out the source element data after it is copied to the target element.
pDeleteTempObj	Numeric	1	Use 0 to retain temporary views and subsets created by the process. Use 1 to delete temporary views and subsets created by the process. Use 2 to delete only the temporary views created by the process.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.Copy',
    'pCube','General Ledger',
    'pViewSource','',
    'pViewTarget','',
    'pDimension','Version',
    'pSourceElement','Budget',
    'pTargetElement','Budget_v2',
    'pSkipRules',0,
    'pZeroTarget',1,
    'pZeroSource',1,
    'pDeleteTempObj',1
    'pDebug',0
    );
```

Bedrock.Cube.Data.Export

Bedrock.Cube.Data.Export

This Bedrock TM1 TI exports data based on a string based filter from the nominated cube to an ASCII file.

Note: If you wish to export a current view use **Bedrock.Cube.Data.ViewExportToFile**

The filter can be based on multiple dimensions and multiple views, here are some standard examples using the standard delimiters:

- **Year: 2010** - Year 2010 only
- **Year: 2010 & Version: Actual** - Year 2010 and version Actual
- **Year: 2010 + 2011** - Year 2010 and Year 2011
- **Year: 2010 + 2011 & Month: Jan + Feb** - Years 2010 and 2011 for the Months of January and February
- **Year: 2010 & Month: All Months** - If the pSkipConsols parameter is set to 1, any consolidated elements passed to the filter will be converted to their N level children. In this case it will be Year 2010 and January, February, March, April, May, etc.

Filter Tips

- Principal or Alias's can be used for elements
- Spaces are ignored
- Consolidated elements are converted to the N level children when pSkipConsols is set to 1 (the default).
- You can use multiple characters for the delimiters, this is important for dimensions were special characters are used. If your dimension has special characters it is suggested that you use multiple character delimiters, i.e.:

  Year:: 2010 ++ 2011 && Version:: Actual

  OR

  Year:= 2010 ++ 2011 && Version:= Actual

  It is rare for multiple special characters to be side by side.

If there is no filter parameter provided, the entire cube will be exported.

Limited to a cube with a maximum of 27 dimensions.

Parameters

pCube	String		The name of the cube where the data exists.
pView	String		The name to use for the temporary view and subsets created by the process. If omitted or blank, a view name consisting of a time stamp and random number is used.
pFilter	String		The filter to be used to create the view to export, see the examples above.
pDimensionDelim	String		The delimiter between dimensions.
pElementStartDelim	String		The delimiter at the end of the dimension name and the start of the list of elements.
pElementDelim	String		The delimiter between elements.
pSkipRules	Boolean	1	Use 0 to include rule calculated data in the copied data. Use 1 to exclude rule calculated data from the copied data.
pSkipCons	Boolean	1	Use 0 to include consolidated data in the copied data. Use 1 to exclude consolidated data from the copied data.
pZeroSource	Boolean	0	Use 1 to zero out the source element data after it is copied to the target element.
pDeleteTempObj	Boolean	1	Use 0 to retain temporary views and subsets created by the process. Use 1 to delete temporary views and subsets created by the process. Use 2 to delete only the temporary views created by the process.
pFilePath	String		The directory where the file is to be saved. If no file path is provided, the ASCII file will be saved to the TM1 logging directory.
pFileName	String		The file name of the ASCII file. If no file name is provided, a combination of the cube, dimension and element suffixed by 'export.csv' will be used.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.ExportToFile',
    'pCube','General Ledger',
    'pView','',
    'pFilter','Year : 2011 + 2012 & Version : Actual' ,
    'pDimensionDelim','&',
    'pElementStartDelim',':',
    'pElementDelim','+',
    'pSkipRules',1,
    'pSkipCons',1,
    'pZeroSource',0,
    'pDeleteTempObj',1,
    'pFilePath','C:\FinancialData',
    'pFileName','Actuals.txt',
    'pDebug',0
    );
```

Bedrock.Cube.Data.ExportToFile

Bedrock.Cube.Data.ExportToFile

This Bedrock TM1 TI exports all the data from a single element within a dimension from the nominated cube to an ASCII file.

Note: If you wish to export a current view use **Bedrock.Cube.Data.ViewExportToFile**

Limited to a cube with a maximum of 27 dimensions.

Parameters

pCube	String		The name of the cube where the data exists.
pDimension	String		The dimension of the source element.
pElement	String		The element in the dimension from which the data needs to be exported.
pFilePath	String		The directory where the file is to be saved. If no file path is provided, the ASCII file will be saved to the TM1 logging directory.
pFileName	String		The file name of the ASCII file. If no file name is provided, a combination of the cube, dimension and element suffixed by 'export.csv' will be used.
pSkipRules	Boolean	1	Use 0 to include rule calculated data in the copied data. Use 1 to exclude rule calculated data from the copied data.
pZeroSource	Boolean	0	Use 1 to zero out the source element data after it is copied to the target element.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.ExportToFile',
    'pCube','General Ledger',
    'pDimension','Version',
    'pElement','Actual',
    'pFilePath','C:\FinancialData',
    'pFileName','Actuals.txt',
    'pSkipRules',1,
    'pZeroSource',0,
    'pDebug',0
    );
```

Bedrock.Cube.Data.ImportFromFile

Bedrock.Cube.Data.ImportFromFile

This Bedrock TM1 TI imports data from a file to a cube.

Limited to a cube with a maximum of 27 dimensions.

Parameters

pSourceDir	String		The directory where the file is saved.
pSourceFile	String		The name of the file to be loaded into the cube.
pCube	String		The name of the cube where the data is to be loaded.
pDimension	String		(Optional) The name of the dimension within which to copy data from source element (exported element) to target element.
pSourceElement	String		(Only required if pDimension is used.) Exported element to copy data from.
pTargetElement	String		(Only required if pDimension is used.) Element to copy data of exported element to.
pTitleRows	Numeric	1	The number of title rows in the file that will be skipped by this Bedrock TM1 TI.
pDelimiter	String	,	The character separator of the data.
pQuote	String	“	The quote character used in the source data.
pAccumulate	Numeric	0	Use 0 to not accumulate amounts when importing. Use 1 to accumulate amounts when importing.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.ImportFromFile',
    'pSourceDir','C:\FinancialData',
    'pSourceFile','Actuals.txt',
    'pCube','General Ledger',
    'pDimension','Version',
    'pSourceElement','Budget',
    'pTargetElement','Budget_v2,
    'pTitleRows',1,
    'pDelimiter',',',
    'pQuote', '"',
    'pAccumulate',0,
    'pDebug',0
    );
```

Bedrock.Cube.Data.ViewExportToFile

Bedrock.Cube.Data.ViewExportToFile

This Bedrock TM1 TI will export the data from a view within the cube. Options to include or exclude consolidated elements, rule calculated data, and null data points can be specified.

Note: If you wish to export all data for a single element in a dimension use **Bedrock.Cube.Data.ExportToFile**

Parameters

pCube	String		The name of the cube where the view exists.
pExportPath	String		The directory where the file is to be saved. If no file path is provided, the ASCII file will be saved to the TM1 logging directory.
pExportFile	String		The file name of the ASCII file. If no file name is provided, a combination of the cube and view suffixed by the word “export.csv” will be used.
pView	String		The name of the view to be exported to the file. If no view is provided then the whole cube will be exported to the file.
pSkipRuleValues	Boolean	1	To include (0) or exclude (1) rule calculated data from the export file.
pSkipCalcValues	Boolean	1	To include (0) or exclude (1) consolidated data from the export file.
pSkipNullValues	Boolean	1	To include (0) or exclude (1) data points that have no data in the export file.

pTitleRecord	Boolean	1	To include (1) or exclude (0) a title row in the export file.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.ViewExportToFile',
    'pCube','General Ledger',
    'pExportPath','C:\FinancialData',
    'pExportFile','2011 Actual Data.txt',
    'pView','2011 Actual Data',
    'pSkipRuleValues',1,
    'pSkipCalcValues',1,
    'pSkipNullValues',1,
    'pTitleRecord',1,
    'pDebug',0
    );
```

Bedrock.Cube.Data.ZeroOut

Bedrock.Cube.Data.ZeroOut

This Bedrock TM1 TI will build a temporary view for a single cube that is to be zeroed out. The temporary view can be restricted by nominating one or more dimensions and elements. Using the filter parameter, nominate the dimension followed by the elements to be included in the temporary subset separated by the Element Delimiter (pDelimElem). If more than one dimension is required, then separate each dimension and element set using the Dimension Delimiter (pDelimDim).

For example: To zero out the current year in the “Year” dimension, four months in the “Month” dimension and the “Actual” element in the Version dimension, the filter parameter value would be:

Version + Actual & Year + 2011 & Month + Sep + Oct + Nov + Dec

If the element is a consolidation in the dimension then all the descendants of that consolidation will be included in the temporary subset. In the example above, October, November and December are members of the “2nd QTR” consolidation. The same filter parameter can therefore be expressed as:

Version + Actual & Year + 2011 & Month + Sep + 2nd QTR

If there is no filter parameter provided, the entire cube will be cleared.

Parameters

pCube	String		The cube name where the view is to be created.
pDelimDim	String	&	Used to distinguish more than one dimension and element set in the pFilter parameter. Change the delimiter to another character if it is used in any dimension or element name in the cube.

pDelimElem	String	+	Used to distinguish more than one element in the pFilter parameter with the dimension element set. Change the delimiter to another character if it is used in any dimension or element name in the cube.
pFilter	String		Restrict the portion of the cube to be cleared by entering dimension and element sets. **For example:** Version + Actual & Year + 2011 & Month + Sep + 2nd QTR
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Data.ZeroOut',
    'pCube','General Ledger',
    'pDelimDim','&',
    'pDelimElem','+',
    'pFilter','Version+Budget',
    'pDebug',0
    );
```

Bedrock.Cube.Delete

Bedrock.Cube.Delete

This Bedrock TM1 TI will delete one or more cubes.

Parameters

pCubes	String		The cube name(s) to be deleted separated by the delimiter. For example: General Ledger & Sales
pDelimiter	String	&	Used to distinguish more than one cube in the pCube parameter. Change the delimiter to another character if it is used in any cube name in the pCube parameter.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Delete',
    'pCubes','Finance & Marketing',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Cube.Dimension.Replace

Bedrock.Cube.Dimension.Replace

This Bedrock TM1 TI replaces an existing dimension with another dimension that exists in the model.

Note: After running this process, all the data in the cube will be lost. The cube rules may need to be modified and re-saved for the cube.

Limited to a cube with a maximum of 27 dimensions.

Parameters

pCube	String		The cube name.
pSourceDim	String		The name of the dimension that is to be replaced. This dimension must be in the specified cube.
pTargetDim	String		The name of the replacement dimension. This dimension must exist in the model.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.Dimension.Replace',
    'pCube','General Ledger',
    'pSourceDim','Business Unit',
    'pTargetDim','Cost Centre',
    'pDebug',0
    );
```

Bedrock.Cube.View.Create

Bedrock.Cube.View.Create

This Bedrock TM1 TI is used to create a view based on a string based filter instead the usual ViewCreate, SubsetCreate & SubsetElementInsert statements. It dramatically simplifies the process of creating views, allowing you to create a simple string and the complexity is done for you. This method is much more readable and reduces the number of lines of code.

The filter can be based on multiple dimensions and multiple views, here are some standard examples using the standard delimiters:

- **Year: 2010** - Year 2010 only
- **Year: 2010 & Version: Actual** - Year 2010 and version Actual
- **Year: 2010 + 2011** - Year 2010 and Year 2011
- **Year: 2010 + 2011 & Month: Jan + Feb** - Years 2010 and 2011 for the Months of January

and February

- **Year: 2010 & Month: All Months** - If the pSkipConsols parameter is set to 1, any consolidated elements passed to the filter will be converted to their N level children. In this case it will be Year 2010 and January, February, March, April, May, etc.

Filter Tips

- Principal or Alias's can be used for elements
- Spaces are ignored
- Consolidated elements are converted to the N level children when pSkipConsols is set to 1 (the default).
- You can use multiple characters for the delimiters, this is important for dimensions were special characters are used. If your dimension has special characters it is suggested that you use multiple character delimiters, i.e.:

  Year:: 2010 ++ 2011 && Version:: Actual

  OR

  Year:= 2010 ++ 2011 && Version:= Actual

  It is rare for multiple special characters to be side by side.

Parameters

pCube	String		The name of the cube that the view will be created on.
pView	String		The name of the view that will be created, the name will also b used for any subsets that are created.
pFilter	String		The filter to be used to create the view, see the examples abov
pSuppressZero	Numeric	1	Skip zero values.
pSuppressConsol	Numeric	1	Skip consolidated values.
pSuppressRules	Numeric	1	Skip rule derived values.
pDimensionDelim	String	&	The delimiter between dimensions.
pElementStartDelim	String	:	The delimiter at the end of the dimension name and the start o the list of elements.
pElementDelim	String	+	The delimiter between each element.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.View.Create',
  'pCube' , 'General Ledger',
  'pView' , 'Archive',
  'pFilter' , 'Year: 2008 + 2009 & Month: Jan + Feb + Mar',
  'pSuppressZero' , 1,
  'pSuppressConsol' , 1,
  'pSuppressRules' , 1,
  'pDimensionDelim' , '&',
  'pElementStartDelim' , ':',
  'pElementDelim' , '+',
  'pDebug' , 0
  );
```

Bedrock.Cube.ViewAndSubsets.Delete

Bedrock.Cube.ViewAndSubsets.Delete

This Bedrock TM1 TI deletes a view and any specified subsets that are attached to the dimensions of the cube. It is used in clean up after Bedrock.Cube.View.Create is used. The process first deletes the view on the cube if it exists and then loops through each dimension on the cube and deletes subsets on those dimensions with the name provided.

Parameters

pCube	String		The name of the cube that the view will be deleted from.
pView	String		The name of the view that will deleted by executing the process.
pSubset	String		The name of the subset(s) that will be deleted by executing the process. If the parameter is blank, any subset(s) named the same as the view will be deleted.
pMode	Numeric	1	Use 0 to retain view and subset(s). Use 1 to delete view and subset(s). Use 2 to delete only the view.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.ViewAndSubsets.Delete',
    'pCube','General Ledger',
    'pView','Archive',
    'pSubset','ArchiveSub',
    'pMode',1,
    'pDebug',0
    );
```

Bedrock.Cube.View.Delete

Bedrock.Cube.View.Delete

This Bedrock TM1 TI will delete views in the nominated cubes. This process is typically used to clean up temporary views as part of an overnight process.

Parameters

pCubes	String		One or more cube names. If this parameter is blank than all cubes will be processed.
pViews	String		The name of one or more views to be deleted from the nominated cubes. Can use wildcard characters.
pDelimiter	String	&	Used to separate multiple cubes or separate multiple views in the above parameters.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.View.Delete',
    'pCubes','General Ledger & Marketing',
    'pViews','Actuals 2011 & Mkt Plan',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Cube.View.Publish

Bedrock.Cube.View.Publish

This Bedrock TM1 TI will publish a private view as a public view.

Parameters

pClient	String		The client name that owns the private view.
pCube	String		The name of the cube where the private view exists.
pView	String		The name of the private view that is to be made public.
pSubPublish	Boolean	1	Make private subsets that are associated with the private view public.
pOverwrite	Boolean	0	Allow the private view to overwrite the public view.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Cube.View.Publish',
    'pClient','Test',
    'pCube','General Ledger',
    'pView','CFO Accounts',
    'pSubPublish',0,
    'pOverwrite',1,
    'pDebug',0
    );
```

Bedrock.Cube.ViewAndSubsets.Create

Bedrock.Cube.ViewAndSubsets.Create

This Bedrock TM1 TI builds a view for a single cube, then assigns empty subsets to that view for one or more dimensions.

Should be used in conjunction with other Bedrock TM1 TI that will define the elements within the subset.

Parameters

pCube	String		The cube name where the view is to be created.
pSuppressZero	Boolean	1	Set whether data points with no data should be suppressed in the view.
pSuppressConsol	Boolean	1	Set whether the consolidations should be suppressed in the view.
pSuppressRules	Boolean	1	Set whether rule calculated data points should be included in the view.
pDimensions	String		The dimensions of the cube where empty subsets are to be assigned to the view. If the subset exists for the dimension then all the elements assigned to the subset will be cleared.
pDelimiter	String	&	Used to separate multiple dimensions in the pDimensions parameter.
pView	String		The name of the view to be created for the cube. If the pView parameter is *null* then the pSubset parameter value is used. If the pSubset parameter is also *null* then the default view name is “}” and the cube name.
pSubset	String		The name of the subset to be created for each dimension nominated in the pDimensions parameter. If the pSubset parameter is *null* then the pView parameter value is used. If pView parameter is also *null* then the default view name is “}” and the cube name.
pDebug	Numeric	0	The Debug mode.

Example

Dimensions

Dimensions

The Bedrock.Dim processes are used for tasks such as importing and exporting dimensions, copying dimensions and working with elements, attributes and hierarchies.

Bedrock.Dim.AllConsols.Delete

Bedrock.Dim.AllConsols.Delete

This Bedrock TM1 TI will delete all consolidation elements within a dimension.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.AllConsols.Delete',
    'pDimension','Account',
    'pDebug',0
    );
```

Bedrock.Dim.AllElements.Delete

Bedrock.Dim.AllElements.Delete

This Bedrock TM1 TI will delete all the elements within a dimension.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.AllElements.Delete',
    'pDimension','Account',
    'pDebug',0
    );
```

Bedrock.Dim.Attr.Delete

Bedrock.Dim.Attr.Delete

This Bedrock TM1 TI will delete an attribute from a dimension.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pAttribute	String		The name of the existing attribute to be deleted.
pDebug	Numeric	0	The debug mode.

Example:

```
ExecuteProcess('Bedrock.Dim.Attr.Delete',
    'pDimension','Account',
    'pAttribute','Account Type',
    'pDebug',0
    );
```

Bedrock.Dim.Attr.ImportFromFile

Bedrock.Dim.Attr.ImportFromFile

This Bedrock TM1 TI will create attributes to a dimension from a file.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pSourceDir	String		The file directory where the data file exists.
pSourceFile	String		The file name that contains the attributes to be loaded.
pTitleRows	Numeric	2	The number of title rows in the file to be skipped
pDelimiter	String	,	The delimiter used in the file.
pQuote	String	“	Quotation character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Attr.ImportFromFile',
    'pDimension','Account',
    'pSourceDir','C:\FileData',
    'pSourceFile','attributes_account_dim.csv',
    'pTitleRows',0,
    'pDelimiter',',',
    'pQuote','"',
    'pDebug',0
    );
```

Bedrock.Dim.Attr.Insert

Bedrock.Dim.Attr.Insert

This Bedrock TM1 TI will insert a new attribute into the dimension. A common use of this process is to avoid opening the Attributes Editor in TM1 if you have a large dimension.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pPrevAttr	String		The existing attribute the new attribute is to be inserted after. Can be left blank.
pAttribute	String		The name of the attribute to be inserted.
pAttributeType	String		The attribute type: S (String), N (Numeric) or A (Alias).
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Attr.Insert',
    'pDimension','Account',
    'pPrevAttr','',
    'pAttribute','Account Type',
    'pAttributeType','S',
    'pDebug',0
    );
```

Bedrock.Dim.Attr.SwapAlias

Bedrock.Dim.Attr.SwapAlias

The Bedrock TM1 TI will swap the principal name of the dimension with an alias.

Parameters

pDimension	String		The name of the dimension on which the process is to run.
pAlias	String		The existing alias that is to become the principal name.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Attr.SwapAlias',
    'pDimension','Account',
    'pAlias','Alias',
    'pDebug',0
    );
```

Bedrock.Dim.Clone

Bedrock.Dim.Clone

This Bedrock TM1 TI will make a copy of an existing dimension.

Parameters

pSourceDim	String		The dimension that is to be cloned.
pTargetDim	String		The name of the dimension to be created. The dimension will be rebuilt if it already exists. If this parameter is null then the source dimension name will be appended with '_Clone'.
pAttr	Boolean	0	Set to 1 to copy the source dimension's attributes to the new dimension.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Clone',
    'pSourceDim','Account',
    'pTargetDim','Account Clone',
    'pAttr',1,
    'pDebug',0
    );
```

Bedrock.Dim.Create

Bedrock.Dim.Create

This Bedrock TM1 TI will create an empty dimension.

Parameters

pDimension	String		The name of the dimension to be created.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Create',
    'pDimension','Line of Business',
    'pDebug',0
    );
```

Bedrock.Dim.Destroy

Bedrock.Dim.Destroy

This Bedrock TM1 TI will destroy an existing dimension as long as it is not part of any cube structure.

Parameters

pDimension	String		The name of the dimension to be destroyed.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Destroy',
    'pDimension','Line of Business',
    'pDebug',0
    );
```

Bedrock.Dim.Element.Create

Bedrock.Dim.Element.Create

This Bedrock TM1 TI will insert a new element into a dimension.

Parameters

pDimension	String		The name of the dimension in which the element is to be inserted.
pElement	String		The principal name of the element to be inserted.

pElementType	String		The element type to be inserted:
			
			N = Numeric;
			
			S = String;
			
			C = Consolidated.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Element.Create',
    'pDimension','Line of Business',
    'pElement','LOB_1',
    'pElementType','N',
    'pDebug',0
    );
```

Bedrock.Dim.Element.Delete

Bedrock.Dim.Element.Delete

This Bedrock TM1 TI will delete an existing element from a dimension.

Parameters

pDimension	String		The name of the dimension in which the element exists.
pElement	String		The principal name or alias name of the element to be deleted.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Element.Delete',
    'pDimension','Line of Business',
    'pElement','LOB_1',
    'pDebug',0
    );
```

Bedrock.Dim.Element.Move

Bedrock.Dim.Element.Move

This Bedrock TM1 TI will either remove an element from a consolidation or assign an element to a consolidation, depending on the chosen action.

Parameters

pDimension	String		The name of the dimension where the element exists.
pElement	String		The principal name or alias name of the existing element to be moved.
pTargetConsol	String		The name of the consolidation that the element is to be either added to or removed from.
pAction	String	Add Delete	Add – Add the element to the consolidation. Delete – Remove the element from the consolidation.
pElWeight	Numeric	1	The weight to be assigned the element within the consolidation.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Element.Move',
    'pDimension','Cost Centre',
    'pElement','Accounting',
    'pTargetConsol','Finance Group',
    'pAction','Add',
    'pElWeight',1,
    'pDebug',0
    );
```

Bedrock.Dim.EmptyConsols.Delete

Bedrock.Dim.EmptyConsols.Delete

This Bedrock TM1 TI deletes any consolidated elements within the dimension that do not have any components. That is, it deletes C level parents that have no children.

Parameters

pDimension	String		The name of the dimension.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.EmptyConsols.Delete',
    'pDimension','Account',
    'pDebug',0
    );
```

Bedrock.Dim.ExportToFile

Bedrock.Dim.ExportToFile

This Bedrock TM1 TI exports the elements of a dimension to a file. The file columns are;

1	Index	The dimension index of the element.
2	Element	The principal name of the element.
3	Alias: [Alias Name]	The first alias of the dimension.
4	Level	The level the element is in the dimension.
5	Parent 1	The first consolidation for the element.
6	Weight 1	The weight of the element to the first consolidation.
7	Parent 2	The second consolidation for the element.
8	Weight 2	The weight of the element to the second consolidation.

9	Parent 3	The third consolidation for the element.
10	Weight 3	The weight of the element to the third consolidation.
11	Parent 4	The fourth consolidation for the element.
12	Weight 4	The weight of the element to the fourth consolidation.
13	Parent 5	The fifth consolidation for the element.
14	Weight 5	The weight of the element to the fifth consolidation.

Parameters

pDimension	String		The name of the dimension.
pExportPath	String		The file path where the file is to be created.
pExportFile	String		The name of the file to be created. If no file name is provided then the file name will be the name of the dimension and '_export.csv'.
pTitleRecord	Boolean	1	1 – The first row of the file will include metadata about the dimension. The second row of the file will be the column headings. 0 – Do not include a header row in the output file.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.ExportToFile',
    'pDimension','Account',
    'pExportPath','C:\Financial Data',
    'pExportFile','Account.txt',
    'pTitleRecord',0,
    'pDebug',0
    );
```

Bedrock.Dim.Hierarchy.Unwind.All

Bedrock.Dim.Hierarchy.Unwind.All

This Bedrock TM1 TI will remove all parent / child relationships in a dimension.

Parameters

pDimension	String		The name of the dimension to be unwound.
pDebug	Numeric		The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Hierarchy.Unwind.All',
    'pDimension','Cost Centre',
    'pDebug',0
    );
```

Bedrock.Dim.Hierarchy.Unwind.Consolidation

Bedrock.Dim.Hierarchy.Unwind.Consolidation

This Bedrock TM1 TI will remove all parent / child relationships below a consolidation in a dimension.

Parameters

pDimension	String		The name of the dimension to be unwound.
pConsol	String		The consolidated element that the elements are to be unwound from.
pRecursive	Boolean	0	0 = Delete only the relationships between the pConsol element and its children. 1 = Delete all the relatiosnips from all the elements under the nominated pConsol consolidation.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Hierarchy.Unwind.Consolidation',
    'pDimension','Cost Centre',
    'pConsol','Finance',
    'pRecursive',1,
    'pDebug',0
    );
```

Bedrock.Dim.ImportFromFile

Bedrock.Dim.ImportFromFile

This Bedrock TM1 TI creates a dimension from an ASCII file (the file format is the same at that created by the **Bedrock.Dim.ExportToFile** Bedrock process)**.** The file columns are;

1	Index	The dimension index of the element.
2	Element	The principal name of the element.
3	Alias: [Alias Name]	The first alias of the dimension.
4	Level	The level the element is in the dimension.
5	Parent 1	The first consolidation for the element.
6	Weight 1	The weight of the element to the first consolidation.
7	Parent 2	The second consolidation for the element.
8	Weight 2	The weight of the element to the second consolidation.
9	Parent 3	The third consolidation for the element.
10	Weight 3	The weight of the element to the third consolidation.
11	Parent 4	The fourth consolidation for the element.
12	Weight 4	The weight of the element to the fourth consolidation.
13	Parent 5	The fifth consolidation for the element.
14	Weight 5	The weight of the element to the fifth consolidation.

Up to five consolidations rollups can be specified in the output file.

Parameters

pSourceDir	String		The directory name where the import file is located.
pSourceFile	String		The file name of the source file to be imported.
pDimension	String		The name of the dimension where the elements will be added.
pAlias	String		The name of the alias to be updated.
pAction	String		Add – The element will be added to the dimension. Replace – All the elements in the dimension will be deleted. New elements will be added from the file.
pTitleRows	Numeric	2	The number of title rows in the import file.
pDelimiter	String	,	The column delimiter.
pQuote	String	“	Quotation character.
pDebug	Numeric	0	The Debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.ImportFromFile',
    'pSourceDir','C:\Financial Data',
    'pSourceFile','Account.txt',
    'pDimension','NewAccount',
    'pAlias','DisplayName',
    'pAction','Add',
    'pTitleRows',0,
    'pDelimiter',',',
    'pQuote','"',
    'pDebug',0
    );
```

Bedrock.Dim.CheckUnusedDimensions

Bedrock.Dim.CheckUnusedDimensions

This Bedrock TM1 TI will output the names of all dimensions that are not assigned to any cubes to a file in the logging directory. The output file name is 'UnusedDimensions.[Time Stamp].csv'.

Parameters

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.CheckUnusedDimensions',
    'pDebug',0
    );
```

Bedrock.Dim.CloneFromSubset

Bedrock.Dim.CloneFromSubset

This Bedrock TM1 TI will make a copy of an existing dimension subset, creating it as a dimension.

Parameters

pSourceDim	String		The dimension that is to be cloned.
pSubset	String		The subset that contains the elements to be cloned.
pTargetDim	String		The name of the dimension to be created. The dimension will be rebuilt if it already exists. If this parameter is null then the source dimension name will be appended with '_Clone'.
pAttr	Boolean	0	Set to 1 to copy the source dimension's attributes to the new dimension.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.CloneFromSubset',
    'pSourceDim','Account',
    'pSubset','All Planning Level'
    'pTargetDim','Account Planning',
    'pAttr',1,
    'pDebug',0
    );
```

Bedrock.Dim.CloneFromSubset.Flat

Bedrock.Dim.CloneFromSubset.Flat

This Bedrock TM1 TI will make a copy of an existing dimension subset, creating it as a dimension. All elements of the source subset will be created as N-level elements in the target dimension.

Parameters

pSourceDim	String		The dimension that is to be cloned.
pSubset	String		The subset that contains the elements to be cloned.
pTargetDim	String		The name of the dimension to be created. The dimension will be rebuilt if it already exists. If this parameter is null then the source dimension name will be appended with '_Clone'.
pAttr	Boolean	0	Set to 1 to copy the source dimension's attributes to the new dimension.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.CloneFromSubset.Flat',
    'pSourceDim','Account',
    'pSubset','All Planning Level'
    'pTargetDim','Account Planning',
    'pAttr',1,
    'pDebug',0
    );
```

Subsets

Subsets

The Bedrock.Sub processes are used for tasks such as creating and editing a dimension's subsets based on attributes and levels in the hierarchy.

Bedrock.Dim.Sub.Create

Bedrock.Dim.Sub.Create

This Bedrock TM1 TI will add elements to a subset based on a number of criteria as follows:-

- Ancestor of a consolidated element
- The attribute value
- The level of the element in the dimension. An upper and lower limit can be specified.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pConsol	String		The consolidated element the element must be a member of to be added to the subset. If this parameter is left blank then all elements will be evaluated.
pAttribute	String		The attribute to be evaluated. If this parameter is left blank then this criteria is not evaluated.
pAttributeValue	String		The value of the attribute the element must equal.
pLevelFrom	String	0	The lowest level in the dimension the element can have.
pLevelTo	String	20	The highest level in the dimension the element can have.
pExclusions	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate multiple elements to be excluded.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create',
    'pDimension','Product',
    'pSubset','Audio Items',
    'pConsol','All Products',
    'pAttribute','Audio/Video',
    'pAttributeValue','Audio',
    'pLevelFrom',0,
    'pLevelTo',1,
    'pExclusions',
    'ViDMP3 & VidAudio',
    'pDelimiter','&',
    'pAddToSubset',0,
    'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.All

Bedrock.Dim.Sub.Create.All

This Bedrock TM1 TI will create a subset with all the elements in a dimension.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate different elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.All',
      'pDimension','product',
      'pSubset','All Items Except Audio and Video',
      'pAddToSubset',0,
      'pExclusions','ViDMP3 & VidAudio',
      'pDelimiter','&',
      'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.Attribute.All

Bedrock.Dim.Sub.Create.Attribute.All

This Bedrock TM1 TI will create a subset based on a value of an attribute. The process evaluates all elements in the dimension including consolidated elements. If the value of the nominated attribute for the element equals the parameter value then the element is added to the subset.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pAttribute	String		The attribute to base the subset on.
pAttributeValue	String		The attribute value of the element must equal to be included into the subset.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate different elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Attribute.All',
      'pDimension','Business Unit',
      'pSubset','All Operational Units',
      'pAttribute','Type of Unit',
      'pAttributeValue','OP',
      'pAddToSubset',0,
      'pExclusions','',
      'pDelimiter','&',
      'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.Attribute.Leaf

Bedrock.Dim.Sub.Create.Attribute.Leaf

This Bedrock TM1 TI will create a subset of leaf elements based on a value of an attribute. This will evaluate only leaf elements in the dimension. If the value of the nominated attribute for the element equals the parameter value then the element is added to the subset.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pAttribute	String		The attribute to base the subset on.
pAttributeValue	String		The attribute value the element must have to be included in

			the subset
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate different elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Attribute.Leaf',
      'pDimension','Business Unit',
      'pSubset','Operational Units',
      'pAttribute','Type of Unit',
      'pAttributeValue','OP',
      'pAddToSubset',0,
      'pExclusions','',
      'pDelimiter','&',
      'pDebug',0
      );
```

Bedrock.Dim.Sub.Create.ByElement

Bedrock.Dim.Sub.Create.ByElement

This Bedrock TM1 TI will add specific elements to a specified subset.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pElements	String		The elements to be added to the subset.
pDelimiter	String	&	The delimiter to separate multiple elements.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.ByElement',
      'pDimension','Business Unit',
      'pSubset','Regional Units',
      'pElements','Western & Northern & Eastern',
      'pAddToSubset',0,
      'pDelimiter','&',
      'pDebug',0
      );
```

Bedrock.Dim.Sub.Create.ByLevel

Bedrock.Dim.Sub.Create.ByLevel

This Bedrock TM1 TI will create a subset for each level of a dimension. The subset will contain the elements for that level. The name of the subset will be 'All Level' and the level of the dimension, for example: 'All Level 01'.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSort	Boolean	0	0 = The subset will be sorted in the same order as the dimension. 1 = Sort the subset alphabetically.
pConvertStatic	Boolean	1	0 = The subset is created by an MDX expression. 1 = The MDX expression will be converted to a static subset.
pDebug	Numeric	1	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.ByLevel',
      'pDimension','Business Unit',
      'pSort',0,
      'pConvertStatic',1,
      'pDebug',0
      );
```

Bedrock.Dim.Sub.Create.ByMDX

Bedrock.Dim.Sub.Create.ByMDX

This Bedrock TM1 TI will create a subset using an MDX expression. This subset can then be converted to a static subset.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pMDXExpr	String		The MDX expression to create the subset

pConvertToStatic	Boolean	1	0 = The subset is created by an MDX expression. 1 = The MDX expression will be converted to a static subset.
pDebug	Numeric	1	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.ByMDX',
      'pDimension',
      'product',
      'pSubset','LA Products',
      'pMDXExpr','{TM1FILTERBYPATTERN( {TM1SUBSETALL( [Product] )}, "*LA-
*")}',
      'pConvertToStatic',1,
      'pDebug',0
      );
```

Bedrock.Dim.Sub.Create.Consolidation.All

Bedrock.Dim.Sub.Create.Consolidation.All

This Bedrock TM1 TI will create a subset of elements that are descendants of the consolidated element.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pConsol	String		The consolidated element that members of the subset must be a descendent of.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate multiple elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Consolidation.All',
      'pDimension','Business Unit',
      'pSubset','By Regions',
      'pConsol','Regional',
      'pAddtoSubset',0,
      'pExclusions','Other',
      'pDelimiter','&',
      'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.Consolidation.Leaf

Bedrock.Dim.Sub.Create.Consolidation.Leaf

This Bedrock TM1 TI will create a subset of leaf (level 0) elements that are descendants of the consolidated element.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pConsol	String		The consolidated element that members of the subset must be descendants of.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate multiple elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Consolidation.Leaf',
      'pDimension','Business Unit',
      'pSubset','By Regions Leaf',
      'pConsol','Regional',
      'pAddtoSubset',0,
      'pExclusions','Other',
      'pDelimiter','&',
      'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.Leaf

Bedrock.Dim.Sub.Create.Leaf

This Bedrock TM1 TI will create a subset with elements that are the leaf level of a dimension.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pAddToSubset	Boolean	0	0 = Add the elements to an empty subset. 1 = Add the elements to an existing subset.
pExclusion	String		Elements to be excluded from the subset.
pDelimiter	String	&	The delimiter to separate different elements to be excluded.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Leaf',
      'pDimension','Business Unit',
      'pSubset','Leaf Elements',
      'pAddtoSubset',0,
      'pExclusions','',
      'pDelimiter','&',
      'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.Orphans

Bedrock.Dim.Sub.Create.Orphans

This Bedrock TM1 TI will create two subsets; 'Orphan C Elements (no children)' and 'Orphan N elements (no parents)'. The 'Orphan C Elements (no children)' subset contains all the consolidation elements that have no component elements. The 'Orphan N elements (no parents)' subset contains all the leaf elements that are not members of any consolidation.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.Orphans',
    'pDimension','Cost Centre',
    'pDebug',0
    );
```

Bedrock.Dim.Sub.Create.TopLevelHierarchy

Bedrock.Dim.Sub.Create.TopLevelHierarchy

This Bedrock TM1 TI will create a subset of the top level consolidations in a dimension. That is, a subset of all the "top node" ancestors which are consolidated C level elements that have no parents.

Parameters

pDimension	String		The name of the dimension where the subset is to be created.
pSubset	String		The name of the subset that the elements will be added to.
pConvertToStatic	Boolean	1	0 = The subset is created by an MDX expression. 1 = The MDX expression will be converted to a static subset.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Create.TopLevelHierarchy',
    'pDimension','Cost Centre',
    'pSubset','Top Level Centres',
    'pConvertToStatic',1,
    'pDebug',0
    );
```

Bedrock.Dim.Sub.Delete

Bedrock.Dim.Sub.Delete

This Bedrock TM1 TI will delete one or more subsets from one or more dimensions.

Parameters

pDimensions	String		The name of the dimension where the subset is to be deleted. Multiple dimensions can be specified by separating the dimension names with the delimiter character.
pSubsets	String	}Bedrock*	The name of the subset to be deleted. Multiple subsets can be deleted by separating the subset names with the delimiter character. Wildcard characters can be used, e.g. 'All Level *'.
pDelimiter	String	&	The delimiter to separate different dimensions or subsets.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Delete',
    'pDimensions','Business Unit',
    'pSubsets','By Regions & Regional Units',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Dim.Sub.Exclude

Bedrock.Dim.Sub.Exclude

This Bedrock TM1 TI will remove specific elements from a subset.

Parameters

pDimension	String		The name of the dimension where the subset exists.
pSubset	String		The name of the subset that the elements are to be excluded from.
pElements	String		The element to be excluded. Multiple elements can be specified by separating the elements using the delimiter character. If a consolidated element is specified then the consolidated element and all its descendants will be excluded from the element.
pDelimiter	String	&	The delimiter to separate multiple elements to be removed.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.Exclude',
    'pDimension','Business Unit',
    'pSubset','Regional',
    'pElements','Other & Unspecific',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Dim.Sub.ExportToFile

Bedrock.Dim.Sub.ExportToFile

This Bedrock TM1 TI will export the members of a subset to a file. The file columns are:-

1	Index	The dimension index of the element.
2	Element	The principal name of the element.
3	Alias: [Alias Name]	The first alias of the dimension.

4	Level	The level the element is in the dimension.
5	Parent 1	The first consolidation for the element.
6	Weight 1	The weight of the element to the first consolidation.
7	Parent 2	The second consolidation for the element.
8	Weight 2	The weight of the element to the second consolidation.
9	Parent 3	The third consolidation for the element.
10	Weight 3	The weight of the element to the third consolidation.
11	Parent 4	The fourth consolidation for the element.
12	Weight 4	The weight of the element to the fourth consolidation.
13	Parent 5	The fifth consolidation for the element.
14	Weight 5	The weight of the element to the fifth consolidation.

Up to five consolidations can be specified in the output file.

Parameters

pDimension	String		The name of the dimension where the subset exists.
pSubset	String		The name of the subset that the elements are to be excluded from.
pExportPath	String		The file directory where the export file will be saved.
pExportFile	String		The name of the export file. If no export file name is provided then the default export file is '[dimension name].[subset name].export.csv'.
pTitleRecord	Boolean	1	1 – The first row of the file will include metadata about the dimension. The second row of the file will be the column headings. 0 – Do not include a header row in the output file.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Dim.Sub.ExportToFile',
    'pDimension','Business Unit',
    'pSubset','Regional',
    'pExportPath','C:\Financial Data',
    'pExportFile','Regional Subset.txt',
    'pTitleRecord',1,
    'pDebug',0
    );
```

Server Administration

Server Administration

Bedrock.Server processes are for tasks that affect the whole TM1 server.

Bedrock.Server.SaveDataAll

Bedrock.Server.SaveDataAll

This Bedrock TM1 TI will perform a 'Save Data All' on the model.

Parameters

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Server.SaveDataAll',
    'pDebug',0
    );
```

Bedrock.Server.DataDir.Backup

Bedrock.Server.DataDir.Backup

This Bedrock TM1 TI will perform a backup of the TM1 data directory.

Parameters

pDataDir	String		The TM1 data directory that is to be copied.
pBackupDir	String		The directory where the copy will be stored.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Server.DataDir.Backup',
    'pDataDir','C:\Program Files\Cognos\TM1\Custom\TM1Data\PlanSamp'
    'pBackupDir','C:\Program Files\Cognos\TM1\Custom\TM1Data\PlanSampBackups'
    'pDebug',0
    );
```

Bedrock.Server.DataDir.ListContents

Bedrock.Server.DataDir.ListContents

This Bedrock TM1 TI will create listings of the contents of the TM1 data directory as text file output in the data directory itself.

Parameters

pDataDir	String		The TM1 data directory to list contents of.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Server.DataDir.ListContents',
    'pDataDir','C:\Program Files\Cognos\TM1\Custom\TM1Data\PlanSamp'
    'pDebug',0
    );
```

Security

Security

Welcome to Bedrock TM1 TI. Bedrock TM1 TI is a collection of Turbo Integrator (TI) processes that are designed to make building models easier and faster than ever before.

Bedrock.Security.Client.Create

Bedrock.Security.Client.Create

This Bedrock TM1 TI can be used to add one or more clients to the model.

Parameters

pClients	String		The name of the client to be added. Multiple clients can be added by separating the clients' names using the delimiter character.
pPassword	String		The TM1 password for the new client.
pMaxPorts	Numeric	5	The maximum number of ports the client is limited to use in the client properties.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Client.Create',
    'pClients','JSmith & BClark',
    'pPassword','abc123',
    'pMaxPorts',5,
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Security.Client.Delete

Bedrock.Security.Client.Delete

This Bedrock TM1 TI can be used to delete one or more clients from the model.

Parameters

pClients	String		The name of the client to be deleted. Multiple clients can be deleted by separating the clients' names using the delimiter character.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Client.Delete',
    'pClients','JSmith & BClark',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Security.Client.Group.Assign

Bedrock.Security.Client.Group.Assign

This Bedrock TM1 TI will assign one or more existing clients to one or more groups or remove one or more existing clients from one or more groups.

Parameters

pClients	String		The name of the client to be added or removed. Multiple clients can be specified by separating the clients' names using the delimiter character.
pGroups	String		The name of the group the clients are to be assigned to or removed from. Multiple groups can be specified by separating the group names using the delimiter character.
pDelimiter	String	&	The delimiter character.
pAddOrRemove	String	Add	Add = Assign the clients to the groups. Remove = Remove the clients from the groups
pSecurityRefresh	String	Yes	Execute a security refresh of the model.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Client.Group.Assign',
    'pClients','JSmith',
    'pGroups','Finance',
    'pDelimiter','',
    'pAddOrRemove','Add',
    'pSecurityRefresh','Yes',
    'pDebug',0
    );
```

Bedrock.Security.Client.Password.Reset

Bedrock.Security.Client.Password.Reset

This Bedrock TM1 TI can be used to reset one or more clients' passwords.

Parameters

pClients	String		The name of the client whose password will be reset. Multiple clients' passwords can be reset by separating the clients' names using the delimiter character.
pPassword	String		The new password for the client.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Client.Password.Reset',
    'pClients','JSmith',
    'pPassword','xyz789',
    'pDelimiter','',
    'pDebug',0
    );
```

Bedrock.Security.ClientGroupSetup

Bedrock.Security.ClientGroupSetup

This Bedrock TM1 TI will create one or more clients and assign those clients to one or more groups.

Parameters

pClients	String		The name of the client to be added. Multiple clients can be added by separating the clients' names using the delimiter character.
pGroups	String		The name of the group to which the clients are to be assigned. Multiple groups can be assigned by separating

			the group names using the delimiter character.
pPassword	String		The TM1 password for the new client.
pMaxPorts	Numeric	5	The maximum number of ports the client is allowed to access in the TM1 model.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Bedrock.Security.Group.Create

Bedrock.Security.Group.Create

This Bedrock TM1 TI will create one or more groups.

Parameters

pGroups	String		The name of the group to be added. Multiple groups can be added by separating the group names using the delimiter character.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Group.Create',
    'pGroups','IT & Marketing',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Security.Group.Delete

Bedrock.Security.Group.Delete

This Bedrock TM1 TI can be used to delete one or more groups.

Parameters

pGroups	String		The name of the group to be deleted. Multiple groups can be deleted by separating the group names using the delimiter character.
pDelimiter	String	&	The delimiter character.

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Group.Delete',
    'pGroups','IT & Marketing',
    'pDelimiter','&',
    'pDebug',0
    );
```

Bedrock.Security.Object.Assign

Bedrock.Security.Object.Assign

This Bedrock TM1 TI can be used to assign one or more groups to an object (application, cube, dimension, process or chore).

Parameters

pGroups	String		The name of the group to be assigned. Multiple groups can be assigned by separating the group names using the delimiter character.
pObjectType	String		The Application, Cube, Dimension, Process or Chore security that is to be changed.
pObjects	String		The Object to which the security is to be applied. Multiple objects can be assigned by separating the object names using the delimiter character.
pSecurityLevel	String		The level of security to be assigned to the groups: Read, Write, Admin or None. Not all security levels are applicable to all objects.
pSecurityRefresh	String	No	Perform a 'Security Refresh' of the model after the objects' security has changed.
pDelimiter	String	&	The delimiter character.
pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Object.Assign',
    'pGroups','Finance',
    'pObjectType','Cube',
    'pObjects','General Ledger',
    'pSecurityLevel','Write',
    'pSecurityRefresh','No',
    'pDelimiter','',
    'pDebug',0
    );
```

Bedrock.Security.Refresh

Bedrock.Security.Refresh

This Bedrock TM1 TI will perform a security refresh on the model.

Parameters

pDebug	Numeric	0	The debug mode.

Example

```
ExecuteProcess('Bedrock.Security.Refresh',
    'pDebug',0
    );
```