IBM Cognos Transformer
Version 10.2.1

UNIX Commands Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 23.

Product Information

This document applies to IBM Cognos Business Intelligence Version 10.2.1 and may also apply to subsequent releases.

Contents

Introduction

This document is intended for use with IBM® Cognos® Transformer when performing modeling and PowerCube-building tasks from the UNIX or Linux command line.

The *IBM Cognos Transformer UNIX Commands Guide* provides the syntax for tasks such as creating and updating PowerCubes and models, publishing PowerCubes, running batch jobs with different preference settings, copying and activating new versions of published PowerCubes, and so on.

For information about modeling and building PowerCubes using the Transformer user interface, see the *IBM Cognos Transformer User Guide*.

Audience

This information is for advanced IBM Cognos Transformer users and IBM Cognos Series 7 cube modelers.

Finding information

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

You can also read PDF versions of the product release notes and installation guides directly from IBM Cognos product disks.

Accessibility features

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.

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.

Cognos Transformer Commands

Transformer can perform certain modeling and cube-building tasks from the command line.

You must invoke the command line utilities from the directory where the Transformer executable resides. In Transformer version 8.x and later, this is the bin directory.

In IBM Cognos Business Intelligence, cogtr replaces the IBM Cognos Series 7 executable, rsserver. However, the basic syntax is the same.

You can combine options using the space character to separate them. However, do not add spaces between an option and its argument, and enclose any intentional spaces found in an argument within double quotes.

In the following example, a Bourne Shell script is appended to the cogtr command to perform a second action (b) on successful completion of the first action (a):

```
#!/bin/sh if cogtr command_line_options then
#perform action a if exit status is 0 else
#perform action b fi
```

This document provides the syntax for the following routine tasks:

- creating or updating cubes
- updating a model with new categories created during the category generation process
- running a set of batch jobs with different preference settings or input files
- changing the current date setting so that relative time calculations in batch cube creation are based on a specific date
- supplying database signon information
- changing the degree of detail for log file messages
- opening and deleting checkpoint files
- verifying and, if necessary, updating the scales used in MDL model columns and measures to match those used in the data source
- publishing cubes in Content Manager
- copying and activating new versions of published cubes
- supporting IBM Cognos prompts
- opening specified .mdl files and executing MDL statements
- specifying the number of records for a test cube
- regenerating categories, and the measure scales used with them, without building the cube

Command Line Syntax

To use command line options, you must start the Transformer executable, cogtr.exe, from the directory in which it is installed.

The syntax for using command line options with optional arguments is as follows:

```
cogtr -option[argument]
```

Use the following guidelines when working with the command line options:

- The option in the command line always starts with a dash (-).
- Command line options are case-insensitive. Arguments are case-sensitive.
- If there are spaces inside any argument, you must enclose the argument in double quotation marks, for example

  ```
  cogtr -k"field three=CarlosR/pw462" Field3.mdl
  ```
- For .py? files, the question mark (?) is replaced by the character that is used in your version of Transformer.
- You can use more than one option in a command line. If an option that is used in a command line is incompatible with an option that appears earlier in the command line, the earlier option is ignored.

Command Line Options

Transformer supports the following command line options. Detailed explanations are provided in the subsections that follow.

Command line options are case-insensitive.

- `-c`; use with `-i`, `-m`, or `-p` to generate categories and creates cubes.
- `-d`; overrides the specified user preference settings
- `-e`; updates the model structure but not the data
- `-f`; specifies the user-defined preference file. Can be used to publish PowerCubes in batch mode and include prompts in an XML command file using the XML schema for preference files.
- `-g`; copies newer versions of cubes to deployment locations and activates the newer versions.

  The command `pcactivate` enables you to activate PowerCubes that are updated in the production environment.
- `-i`; opens the specified .py? model and restarts a failed process from the beginning. Cannot be used with `-s`.
- `-j`; publishes a single PowerCube or all enabled PowerCubes in a cube group
- `-h`; displays help for the command line
- `-k`; specifies database signon information for IBM Cognos Series 7
- `-l`; specifies user authentication information for IBM Cognos BI environment
- `-m`; opens the specified .mdl file or accepts Model Definition Language (MDL) statements
- `-o`; turns off various model and cube creation actions
- `-p`; opens the specified binary model file, .py?, where the question mark (?) is replaced by the character that is used in your version of Transformer. Not valid with an MDL file.
- `-r`; specifies the level of detail for error-logging reports. Valid levels are 0, 1, 2, 3, and 4.
- `-s`; saves the model. Cannot be used with `-i` or `-p`.
- `-t`; sets the current period
- `-u`; gets the partition status for previously generated cubes. Cannot use this option with secured cubes.
- `-v`; use with `-c`, `-m`, or `-p` to specify the number of records for the test cube
- `-x`; updates the column and measure scales based on the data source

- -y; specifies how IBM Cognos Series 7 user-class security conversion is performed.
  - Use -y1 to preserve both the IBM Cognos Series 7 user classes and custom views associated with the IBM Cognos Series 7 model.
  - Use -y2 to preserve only the custom view associated with the IBM Cognos Series 7 model.
  - Use -y3 to discard the IBM Cognos Series 7 user classes and custom views associated with the IBM Cognos Series 7 model.

-c option

This option loads a model file, interprets MDL statements, generates categories, and creates cubes.

Use this option with the applicable file-opening option: either -p, -m, or -i.

The basic syntax for using the -c option is as follows:

```
cogtr -cpfilename.py?|-mfilename.mdl
```

The following example uses -c and -p options together to open the binary model file go_sales.pyj and process it as described.

```
cogtr -c -pgo_sales.pyj
```

The following example uses the -c and -m options together to open the equivalent full model definition (the .mdl text file for go_sales), and process it as described.

```
cogtr -c -mgo_sales.mdl
```

-d option

This option sets a new value for a Transformer user preference. The value overrides settings from the Preferences property sheet for this instance only.

No space may appear between the -d option and its argument. The argument is case-sensitive and must match the value specified in the cogtr.xml file.

For example, use -dLogFileName not -dlogfilename. The basic syntax for using the -d option is as follows:

```
cogtr -dpreference_var=
setting -pfilename.py?|-m
filename.mdl
```

If you specify the -d option after the -f option, the -d setting overrides the setting for -f. The reverse is also true; the last-appearing option overrides the options that precede it.

You can set preferences in the command line, environment variables, and cogtr.xml. The priority of settings is that the setting in the command line takes precedence over the setting in the environment variable, and the setting in the environment variable takes precedence over the setting in cogtr.xml. If you set a preference more than once in a command line, the last setting overwrites all previous settings.

You can use most settings in the cogtr.xml file as arguments for *preference_var* .

You can set global preferences using environment variables.

There is a sample preferences file, cogtr.xml.sample in the *installation_location*/configuration directory. The actual preferences file, cogtr.xml, is not installed by default. It is created and saved to the *installation_location*/configuration directory the first time you save changes to the **Preferences** property sheet in Microsoft Windows. For more information, see the topic, "cogtr.xml File Settings," in the Transformer *User Guide*.

The following example overrides the preference file setting for the DataSourceDirectory, changing it to C:\Newdata.

```
cogtr -dDataSourceDirectory=C:\Newdata -mTransact.mdl
```

The following example overrides the default value at which a warning is issued, for a parent category having too many descendants. The new preference setting (threshold) is 25 children.

```
cogtr -dChildRatioThreshold=25 -mTransact.mdl
```

Preference Settings or Environment Variables

You can also use environment variables to set preferences. Transformer recognizes the following preference settings or environment variables.

For more information, see the topic "Controlling Processing with Preference Settings or Environment Variables" in the *IBM Cognos Transformer User Guide*.

- CenturyBreak
  Default: 20
- ChildRatioThreshold
  Default: 35
  Minimum: 1
  Maximum: 4294967295
- CubeSaveDirectory
  Default: The temp subdirectory in the *installation_location*/c10 directory
- DataSourceDirectory
  Default: The data subdirectory in the *installation_location*/c10 directory
- DataWorkDirectory
  Default: The temp subdirectory in the *installation_location*/c10 directory
- DecimalPoint
  Default: A period (.)
- DefaultSeparator
  Default: A comma (,)
- FilenameVariables
  Default: FALSE
  This setting is not supported on Windows.
- IncUpdateWarning
  Default: TRUE
- LogDetailLevel
  Default: 4
  Minimum: 0
  Maximum: 4
- LogFileAppend
  Default: FALSE

- LogFileDirectory

  Default: The logs subdirectory in the *installation_location*/c10 directory. If the logs subdirectory does not exist, then the default is the current working directory.
- LogFileName

  Default: ""
- LoggingFrequency

  Default: -1
- LunarFiscalLabeling

  Default: TRUE
- MaxTransactionNum

  Default: 50000
- ModelSaveDirectory

  Default: The temp subdirectory in the *installation_location*/c10 directory
- ModelWorkDirectory

  Default: The temp subdirectory in the *installation_location*/c10 directory
- MultiFileCubeThreshold

  Default: 0 (disabled)

  Minimum: 0

  Maximum: 4294967295
- OrderByCategoryLabeling

  Default: 0 (disabled)
- PPDS_READ_MEMORY

  Default: 8000000

  Minimum: 1000000
- PPDS_WRITE_MEMORY

  Default: 4000000

  Minimum: 1000000
- ThousandSeparator

  Default: a comma (,)
- WorkFileMaxSize

  Default: 1500000000

  Minimum: 100000000

  Maximum: 1500000000

-e option

This option updates and saves all the cube metadata that is defined in the model, but does not update the data. The cube metadata consists of object names, labels, short names, descriptions, drill-through reports, and security information.

You cannot use this option with `-c`, and you should always use it in combination with the `-o` option.

The basic syntax for using the `-e` option is as follows:

```
cogtr -e -pfilename.py?|-m
filename.mdl
```

The following example opens the go_sales.pyj model file and updates the defined cubes without regenerating existing categories or creating new ones. It then saves

the model file along with its updated cube metadata: that is, the object names, labels, descriptions, drill-through reports, and security information.

```
cogtr -e -o -pgo_sales.pyj
```

-f option

This option specifies the user-defined preference file or files to use. If you do not include the full directory path with the file name, Transformer searches the executable directory of your most recently installed rendition of the product for the required .xml file.

This option is also used to include prompts in an XML command file using the XML schema for preference files "XML Schema for Preference Files," and to publish PowerCubes in batch mode "Publishing in Batch-mode" on page 8.

If you are performing batch tasks that require the use of multiple preference files, Transformer combines the settings in each file successively; later settings override previously defined ones.

Similarly, if you specify the `-f` option after the `-d` option, the `-f` setting overrides the setting for `-d`. The reverse is also true; the last-appearing option overrides the options that precede it.

No space may appear between the `-f` option and its argument, *preference_file* .

Tip: You can base your preference file entries on the settings the cogtr.xml file, and then run batch jobs by referencing the appropriate file using the `-f`option. Set your environment variables globally or use the `-d`option for specific command sequences.

The basic syntax for using the `-f` option is as follows:

```
cogtr -fpreference_file -p
filename.py?|-mfilename.mdl
```

The following example sets the preference file to C:\Monthly.xml, opens the model file Transact.mdl, then runs the process in batch to create all of the cubes defined in the model:

```
cogtr -fc:\Monthly.xml -mTransact.mdl
```

The following example parses the go_sales.mdl model file using mypref.prf as the specified preference file:

```
cogtr -fmypref.prf -mgo_sales.mdl
```

XML Schema for Preference Files

The XML file format supports multi-value commands and user-defined preference files. Commands in XML files are executed sequentially unless specific rules are defined.

Example

The XML schema can have multiple sections and multiple commands. Commands can contain multiple parameters and parameters can have multiple values.

The cogtr.xml file conforms to the XML schema. The cogtr.xml file contains two major predefined groups of xml elements:

- a section which contains a list of preferences, for example:

```
<Sections>
<Section Name="Transformer">
<Preference Name="DataWorkDirectory" Value="..\temp"/>
<Preference Name="AutoEdit"Value="0"/>
<Preference Name="ChildRatioThreshold" Value="35"/>
<Preference Name="CubeSaveDirectory" Value="..\temp"/>
<Preference Name="DataSourceDirectory" Value="..\temp"/>
</Section><Section Name="RecentFileList">
<Preference Name="File1" Value="NationalOriginal.mdl"/>
<Preference Name="File2" Value="Cubexx.mdl"/>
<Preference Name="File3" Value="GreatOutdoorsCompany_Slow_v1.mdl"/>
<Preference Name="File4" Value="testcube.mdl"/>
</Section>
</Sections>
```

- a Commands section, which will be empty in most cases. The Commands section passes commands to Transformer when it is used in batch mode.

The following example shows multiple preferences and commands being passed to Transformer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 sp2
 (http://www.altova.com)-->
<Settings xsi:noNamespaceSchemaLocation="cogtr_format_v2.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Sections>
  <Section Name="Transformer">
   <Preference Name="DataWorkDirectory" Value="..\temp"/>
   <Preference Name="AutoEdit" Value="0"/>
   <Preference Name="ChildRatioThreshold" Value="35"/>
   <Preference Name="CubeSaveDirectory" Value="..\temp"/>
   <Preference Name="DataSourceDirectory" Value="..\temp"/>
  </Section>
  <Section Name="RecentPackageList">
   <Preference Name="mru_entry_0" Value="/content/package[@name=&apos;EquifaxCube&apos;]"/>
  </Section>
  <Section Name="RecentFileList">
   <Preference Name="File1" Value="c:\NationalOriginal.mdl"/>
   <Preference Name="File2" Value="c:\Modified Cubexx.mdl"/>
   <Preference Name="File3" Value="c:\GreatOutdoorsCompany_Slow_v1.mdl"/>
   <Preference Name="File4" Value="c:\testcube.mdl"/>
  </Section>
 </Sections>
 <Commands>
  <Command Name="Publish">
   <Parameters>
    <Parameter Name="CubeName" Value="test"/>
    <Parameter Name="CognosConnectionDataSourceName" Value="test"/>
    <Parameter Name="DataSourceWindowsLocation"
     Value="c:\test1.mdc"/>
    <Parameter Name="DataSourceUnixLinuxLocation" Value=""/>
    <Parameter Name="DataSourceNameSpace" Value=""/>
    <Parameter Name="ReadCacheSize" Value="0"/>
    <Parameter Name="DataSourceSignon" Value="FALSE"/>
    <Parameter Name="DataSourceDescription" Value=""/>
    <Parameter Name="DataSourceToolTip" Value=""/>
    <Parameter Name="DataSourceUpdate" Value="FALSE"/>
    <Parameter Name="PackageName" Value="tpc"/>
    <Parameter Name="Packagelocation" Value="/content"/>
    <Parameter Name="PackageDescription" Value=""/>
    <Parameter Name="PackageToolTip" Value=""/>
```

```
    <Parameter Name="PackageUpdate" Value="FALSE"/>
    <Parameter Name="PackageAllowNullSuppression" Value="TRUE"/>
    <Parameter Name="PackageAllowMultiEdgeSuppression"
     Value="TRUE"/>
    <Parameter Name="PackageAllowAccessToSuppressionOptions"
     Value="TRUE"/>
   </Parameters>
  </Command>
  <Command Name="prompts" Value="MyQuery1">
   <Parameters>
    <Parameter Name="Prompt1" Value="TestCube"/>
    <Parameter Name="Prompt2">
     <Values>
      <Value>c:\test\cube.mdc</Value>
      <Value>c:\test\cube1.mdc</Value>
      <Value>c:\test\cube2.mdc</Value>
     </Values>
    </Parameter>
    <Parameter Name="Prompt3" Value="TestPackage"/>
   </Parameters>
  </Command>
  <Command Name="prompts" Value="MyQuery2">
   <Parameters>
    <Parameter Name="Prompt4">
     <Values>
      <Value>1</Value>
      <Value>4</Value>
      <Value>8</Value>
     </Values>
    </Parameter>
   </Parameters>
  </Command>
 </Commands>
</Settings>
```

Publishing in Batch-mode

The -f command can be used to publish PowerCubes in batch mode.

Example

The syntax is

```
cogtr -fspec_file.xml -o
-mfilename.mdl
```

where spec_file.xml represents the path and name of the publish specification file, and where `-o` disables the generation of categories and cube creation.

The following table describes the parameters specified in the publish specification file.

- `CubeName`

  Specifies the name of the PowerCube in the model. This is a mandatory value.

- `CognosConnectionDataSourceName`

  Specifies the name of the data source. This is a mandatory value. The set will fail if it is not defined.

  Default: The name in the model.

- `DataSourceWindowsLocation`

  Specifies the Windows location of the data source.

  Default: The current cube location on Windows.

- `DataSourceUnixLinuxLocation`

Specifies the UNIX and Linux location of the data source.

Default: Empty

- `ReadCacheSize`

  Specifies the read cache size for PPDS.

  Default: 0

- `DataSourceNameSpace`

  Specifies the authentication namespace for the data source.

  Default: Empty

- `DataSourceSignon`

  Specifies the command used to create the data source signon, if needed.

  Default: False

- `DataSourceDescription`

  Specifies the description of the data source.

  Default: Empty

- `DataSourceToolTip`

  Specifies the tool tip for the data source.

  Default: Empty

- `DataSourceUpdate`

  Specifies the command used to update a data source.

  Default: FALSE

- `PackageName`

  Specifies the name of the package in Content Manager. This is a mandatory value. The set will fail if it is not defined.

  Default: The name in the model.

- `Packagelocation`

  Specifies the location of the package.

  Default: Public Folders

- `PackageDescription`

  Specifies the description of the package.

  Default: Empty

- `PackageToolTip`

  Specifies the tool tip for the package.

  Default: Empty

- `PackageUpdate`

  Specifies the command used to update the existing package when its settings change.

  Default: FALSE

- `PackageAllowNullSuppression`

  Specifies the command used to specify whether suppression is available to IBM Cognos studio users when working with this package.

  Default: TRUE

- `PackageAllowMultiEdgeSuppression`

  Specifies the command used to specify whether IBM Cognos studio users can suppress both rows and columns. If this option is set to FALSE, users can suppress rows only or columns only. If this parameter is set to TRUE, the PackageAllowNullSuppression parameter must also be set to TRUE.

Default: TRUE

- `PackageAllowAccessToSuppressionOptions`

  Specifies the command used to specify whether IBM Cognos studio users can control the types of empty values that will be suppressed, such as zero or missing values. Types of empty values that users can choose to suppress depend on the studio. If this parameter is set to TRUE, the PackageAllowNullSuppression parameter must also be set to TRUE.

  Default: TRUE

The following example of a publish specification file shows the xml code for publishing a cube.

```
<Command Name="Publish">
 <Parameter Name="CubeName" Value="NATIONAL"/>
 <Parameter Name="CognosConnectionDataSourceName" Value="NATIONAL"/>
 <Parameter Name="DataSourceWindowsLocation" Value="c:\test\cube.mdc"/>
 <Parameter Name="DataSourceUnixLinuxLocation" Value=""/>
 <Parameter Name="DataSourceNameSpace" Value=""/>
 <Parameter Name="ReadCacheSize" Value="0"/>
 <Parameter Name="DataSourceSignon" Value="FALSE"/>
 <Parameter Name="DataSourceDescription" Value=""/>
 <Parameter Name="DataSourceToolTip" Value=""/>
 <Parameter Name="DataSourceUpdate" Value="FALSE"/>
 <Parameter Name="PackageName" Value="NATIONAL"/>
 <Parameter Name="Packagelocation" Value=""/>
 <Parameter Name="PackageDescription" Value=""/>
 <Parameter Name="PackageToolTip" Value=""/>
 <Parameter Name="PackageUpdate Value="FALSE"/>
 <Parameter Name="PackageAllowNullSuppression" Value="TRUE"/>
 <Parameter Name="PackageAllowMultiEdgeSuppression" Value="TRUE"/>
 <Parameter Name="PackageAllowAccessToSuppressionOptions" Value="TRUE"/>
</Command>
```

Including Transformer Prompts in an XML Command File

You can include prompts in an XML command file. You must use the following command line in order for Transformer to read the file:

```
cogtr -f command file name
```

where *command file name* contains a sequence of statements that define prompt values.

The command file can contain one or more commands for prompts. The command name is `prompt`. The value attribute of the command specifies the prompt name. Each `prompt` command contains one or more `Parameter` elements that specify a query name, prompt attributes and values. The Query `Parameter` element specifies the query to which the prompt belongs. The other `Parameter` elements define the prompt type and values.

There are several different types of prompts: simple, multi-valued, range, and MUN.

The following example shows single value, multi-value and range prompts.

```
<?xml version="1.0" encoding="UTF-8"?>
<Settings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Commands>
 <!-- SINGLE_VALUE PROMPT -->
  <!-- Prompt name -->
  <Command Name="prompt" Value="MyPrompt1">
   <Parameters>
    <!-- A query this prompt belongs to -->
```

```
    <Parameter Name="Query" Value="Promptmany~1"/>
    <!-- SingleValue, MultiValue, Range -->
    <Parameter Name="PromptType" Value="SingleValue"/>
    <!-- Any type that understood by RS. Optional. Not in use -->
    <Parameter Name="PromptValueType" Value ="Integer"/>
    <!-- Value -->
    <Parameter Name="PromptValue" Value="12345"/>
   </Parameters>
  </Command>
 <!-- MULTI_VALUE PROMPT -->
  <!-- Prompt name -->
  <Command Name="prompt" Value="MyPrompt2">
   <Parameters>
    <!-- A query this prompt belongs to -->
    <Parameter Name="Query" Value="Promptmany~1"/>
    <!-- SingleValue, MultiValue, Range -->
    <Paramete Name="PromptType" Value="MultiValue"/>
    <!-- Any type that understood by RS. Optional. Not in use -->
    <Parameter Name="PromptValueType" Value ="String"/>
    <Parameter Name="Values">
     <Values>
      <Value>abc</Value>
      <Value>bcd</Value>
      <Value>cdf</Value>
      <Value>dfe</Value>
      </Values>
    </Parameter>
   </Parameters>
  </Command>
 <!-- RANGE -->
  <!-- Promptname -->
  <Command Name="prompt" Value="RangePrompt">
   <Parameters>
    <!-- A query this prompt belongs to -->
    <Parameter Name="Query" Value="Promptmany~1"/>
    <!-- SingleValue, MultiValue, Range -->
    <Parameter Name="PromptType" Value="Range"/>
    <!-- Any type that understood by RS. Optional. Not in use -->
    <Parameter Name="PromptValueType" Value ="String"/>
    <Parameter Name="Values">
     <Values>
      <Value>Range - form</Value>
      <Value>Range - to</Value>
     </Values>
    </Parameter>
   </Parameters>
  </Command>
 </Commands>
</Settings>
```

Simple Prompt

A simple prompt is a variable that has a single value. For example

```
<Parameter Name="SimplePrompt" Value="Single Value"/>
```

Multi-valued Prompt

A multi-valued prompt is a variable that has a number of different values. For example

```
<Parameter Name="Multi-valued-variable">
 <Values>
   <Value>Value1</Value>
```

```
   <Value>Value2</Value>
   <Value>Value3</Value>
 </Values>
</Parameter>
```

Range Prompt

A range prompt is a type of multi-valued prompt. It contains a Range-from and a Range-to value. For example

```
<Parameter Name="Range-variable">
 <Values>
    <Value>Range-from</Value>
    <Value>Range-to</Value>
 </Values>
</Parameter>
```

A range prompt does not need to contain both Range-from and Range-to values, but it must have one or the other, as shown below.

```
<Parameter Name="Range-variable">
 <Values>
   <Value/>
   <Value>Range-to</Value>
 </Values>
</Parameter>
```

MUN Prompt

A MUN prompt is a type of single-value prompt. It must refer to a Member Unique Name, for example

```
<Parameter Name="[ASPV]" Value="[AMERICA].[WORLD]"/>
```

-g option

This option copies newer versions of cubes to deployment locations and activates the newer versions.

If you want to create the newer version of the cube and then copy and activate it, use this -g option with the `-c` option.

The syntax for using the `-g` option is as follows:

```
cogtr -g[powercube_name]|[
powercube_group_name/child_cube_name] -p
filename.py?|-mfilename.mdl
```

The *powercube_name* parameter represents the name of the cube in the Transformer model. If *powercube_name* specifies a cube group that is not a time-based partitioned cube, all child cubes of the cube group are copied to the deployment location and activated. If *powercube_name* specifies a time-based partitioned cube, it is processed as a single cube; member cubes are not deployed individually.

If a name conflict exists with a child cube of a cube group that has the same name as another cube in the model, use the *powercube_group_name/child_cube_name* parameters.

The following example copies and activates all cubes.

```
cogtr -g
```

The following example copies and activates the child cube named "France" in a cube group named "Europe".

```
cogtr -gEurope/France
```

-h option

This option displays help for the command line. Help is also displayed if you do not provide any command options.

The basic syntax for using the `-h` option is as follows:

```
cogtr -h
```

-i option

This option opens a saved model, regardless of the existence of a checkpoint file.

You cannot use the `-i` option with the `-s` option.

A checkpoint file is automatically created when a model is suspended due to a general system failure, such as that caused by a power outage. The next time the model is opened in interactive mode, users can open either the checkpoint file or the last-saved version of the model file. In batch mode, you can use the `-i` option to bypass the prompt and force Transformer to open the original model file instead of the checkpoint file.

The basic syntax for using the `-i` option is as follows:

```
cogtr -i -pfilename.py?
```

Note: Checkpoint files have a .qy? extension. As with .py? files, the ? (question mark) in the extension is replaced by the character that is used in your release of Transformer, such as .qyj.

The following example opens the model file, Sales.py?, discarding any existing checkpoint file, then runs the process in batch mode to create all defined cubes:

```
cogtr -i Sales.py?
```

-j option

This option is used to publish a single PowerCube or all enabled PowerCubes in a cube group. Both the data source connection and the package are published for a particular PowerCube.

The syntax for this option is as follows:

`cogtr -j` *`cube_name`* or `cogtr -ju` *`cube_name`*

where *cube_name* represents the PowerCube name or the cube group name. If no name is specified, all cubes in the model are published except for the time-based partitioned cubes where only the parent cube needs to be published.

The -j form of this option publishes only new PowerCubes and keeps the previously published cubes unchanged. A data source connection for the PowerCube is published if a connection with the same name does not already exist. If the data source connection is successfully published, the associated package is also created if a package with the same name does not already exist.

The -ju form of this option updates previously published PowerCubes and cube groups in addition to publishing PowerCubes that were not previously published.

The following example shows how to publish a cube or a cube group named National. The cube is built and the categories are generated.

```
cogtr -j National -c -mNational.mdl
```

The following example shows how to publish one child cube named MA in a cube group named National. The categories and cubes in the model are not generated.

```
cogtr -j National/MA -mNational.mdl
```

The following example shows how to update the data sources or packages associated with the MA and CA cubes in the cube group named National.

```
cogtr -ju National/MA -ju National/CA -mNational.mdl
```

-k option

This option supplies the signon information needed to establish one or more database connections during batch processing.

It provides an alternative to storing the information in the **Signons** list or retrieving user IDs and passwords from the configured authentication source. This is especially useful when used with .mdl files which, by default, do not use verb MDL and so do not store signon passwords.

For databases referenced in an Impromptu® Query Definition (IQD) file, the signon is the logical database name that appears in the related Transformer .xml file. Multiple IQD data sources can use the same signon object.

Tip: You can view these signons on the Transformer (Windows) interface, but you cannot change them.

The basic syntax for using the -k option is as follows:

```
cogtr -ksignon=
userid/password -p
filename.py?|-mfilename.mdl
```

No spaces may appear between -k and its argument, and the signon name cannot be empty. Also, the signon name cannot contain the ASCII equals (=) character, and the user ID cannot contain a forward slash (/), because both characters are reserved for the parameter syntax.

The following example reads the data source for model Xyzsales.mdl from an Oracle database using an IQD file, confirms the security information, and processes the model in batch mode. The signon sal_log includes the Oracle user ID corpadm and the password my_pass. The command to process the .mdl file for model Xyzsales.mdl is as follows:

```
cogtr -c -s -mXyzsales.mdl -ksal_log=corpadm/my_pass
```

When you use this command, the user ID and password appear in plain text. For information about how to avoid plain text passwords, see the topic "Avoiding Plain Text Passwords When Using the -k or -l Option."

Avoiding Plain Text Passwords When Using the -k or -l Option

If you use the -k or -l option to pass user IDs and passwords to the Transformer command line, you must take precautions to avoid plain text passwords.

Breaches can occur if unauthorized users are able to invoke the ps command or can gain access to the detailed processing information in the log file.

To avoid plain text passwords, adopt one of the following strategies:

- Add MDL statements to the end of the .mdl model file that embed the user ID and password information and update the signon information needed to log on to the database. Then, run cogtr using the -m option, specifying your modified .mdl file.

  For example, embed the authentication information using the following MDL statements:

  ```
  SignonUpdate "sal_cube" PromptForPassword False UserID "corpis" Password
  "bld_cube"
  ```

- Create a secured but temporary MDL script on the server to update the model signon. Then, run cogtr using the -m option, specifying your modified .mdl file.

  For example, create the following MDL script, for batch processing using the -m option:

  ```
  OpenPY "Xyzsales.pyj"SignonUpdate "sal_cube" PromptForPassword False Password
  "bld_cube"SavePY "Xyzsales.pyj"
  ```

For more information about security, see the *Administration and Security Guide*.

-l option

This option specifies the user authentication information for IBM Cognos.

It supplies the signon information needed to authenticate users in one or more security namespaces. The -l option provides an alternative to storing the information in the Signons list.

The basic syntax for using the -l option is as follows:

```
cogtr -lsignon=userid/
password -pfilename.py?|-m
filename.mdl
```

The following example opens the Field3.mdl file and grants access to the required namespace using the signon named field, user ID CarlosR, and password pw462. Assuming that the signon matches what is defined in the model, the process runs in batch mode, creating the cubes as defined in the model.

```
cogtr -lfield=CarlosR/pw462 -mField3.mdl
```

IBM Cognos BI software can be configured to use authentication to an external namespace where users are prompted for credentials as part of the logon process.

You can create signons to build cubes in batch mode in the IBM Cognos environment. Those signons maintain the user ID, password, and the associated namespace name. You can create as many signons as the number of namespaces that your users need to log on to in IBM Cognos BI environment. For Transformer to use the signon automatically, enable the **Set As Auto Logon** property.

If multiple signons with the **Set As Auto Logon** property checked exist in a saved model, when using the command line you do not need to specify the –l option because Transformer will authenticate against all existing signons by default.

When you use this command, the user ID and password appear in plain text. For information about how to avoid plain text passwords, see the topic "Avoiding Plain Text Passwords When Using the -k or -l Option" on page 14.

-m option

This option specifies the .mdl-format model or script file to be processed.

If you use multiple occurrences of `-m`, files are processed in the order of their occurrence.

The basic syntax for using the `-m` option is as follows:

```
cogtr -mfilename.mdl
```

The following example lists the steps for processing a sample model using the `-m` option:

1. Create a separate file, Savemdl.mdl, that contains the line `SaveMDL "Xyznew.mdl"`.
2. For the file Xyzsales.mdl, use the following command:
   ```
   cogtr -mXyzsales.mdl -mSavemdl.mdl
   ```
3. For the file Xyzsales.pyj, use the following command:
   ```
   cogtr -pXyzsales.pyj -mSavemdl.mdl
   ```

-o option

This option loads the model file, but turns off population of the model with data and cube creation.

The basic syntax for using the `-o` option is as follows:

```
cogtr -o -pfilename.py?|-m
filename.mdl
```

The following example loads the file, go_sales.mdl, but disables both population of the model and creation of cubes:

```
cogtr -o -mgo_sales.mdl
```

-p option

This option opens the checkpoint file or loads a binary model file and processes it, beginning from the last checkpoint saved in the checkpoint file if this exists, or beginning from the start of the .py? file. All changes are saved on termination.

This option is not valid with an .mdl file.

Note: A checkpoint file exists if model processing was suspended. The file has a .qy? extension, where the question mark (?) is replaced by the character that is used in your version of Transformer.

The basic syntax for using the `-p` option is as follows:

```
cogtr -pfilename.py?
```

The following example starts Transformer, process the MDL verb commands in the file monthly_update.mdl, obtains preference settings from the file, trnsfrm_prd.xml, and saves the model:

```
cogtr -pgo_sales_jan.pyj -mmonthly_update.mdl -ftrnsfrm_prd.xml
```

-r option

This option sets the degree of detail for messages written to the log file.

Each level includes the errors and messages for the higher levels. No spaces may appear between `-r` and its `log_level` argument, which is assigned a value from 0 to 4, as follows:

- 0 - the Enable Message Logging box is cleared; logging is suppressed
- 1 - includes only severe errors
- 2 - includes error messages and level 1 messages
- 3 - includes warning messages, and level 1 and 2 messages
- 4 - includes all message levels, from informational to severe; the default setting

The basic syntax for using the `-r` option is as follows:

```
cogtr -rlog_level -p
filename.py?|-mfilename.mdl
```

The following example opens the model file Roofing.mdl, sets the degree of detail for messages to 2, then runs the process in batch mode to create all defined cubes. The messages are written to F:\Test\Roof.log.

```
cogtr -r2 -dLogFileName=Roof.log -dLogFileDirectory=F:\Test Roofing
```

-s option

This option, after successful creation of a cube, saves the model with any new categories added during the category generation process, then closes Transformer.

Do not use this option with `-i` or `-p`.

The basic syntax for using the `-s` option is as follows:

```
cogtr -mfilename.mdl -s
filename.py?|filename.mdl
```

where parameters for `-s` are optional.

The following example starts Transformer, parses a text model file (.mdl), and saves the changes in a binary model file (.py?).

```
cogtr -mgo_sales.mdl -sgo_sales.pyj
```

-t option

This option sets the current period for the purpose of calibrating relative time calculations.

It is equivalent to manually defining a current period on the Windows interface, after clearing the **Automatically Set Current Time Period** box on the **Time** tab of the **Dimension** property sheet.

No spaces may appear between `-t` and its argument, and if the category contains hyphens or space characters (as in the sample date range below), you must enclose it in double quotation marks. Also, the `category_code` portion of the command is case-sensitive. This identifier must exactly match the category code in the model.

The basic syntax for using the `-t` option is as follows:

```
cogtr -tcategory_code -p
filename.py?|-mfilename.mdl
```

The following example opens the model Year3.mdl, sets the current period to the category which has a category code of 20061201-20061231, then runs the process in batch mode to create all defined cubes.

```
cogtr -t"20061201-20061231" -sYear3.mdl
```

-u option

This option writes the partition information for a specified cube to the log file, using the following format:

```
date
time  Cube [
powercube_name]|[powercube_group_name/
child_cube_name] partition report Partition # Category
Code Category Name Partition Size
```

You must generate cubes before their partition status can be reported. If the command line includes options to generate categories and create cubes, those options are processed before the partition information is obtained.

Partition information cannot be provided for cubes and cube groups to which security has been applied. To obtain the partition information of cubes in a cube group, you must specify the individual cube names, not the cube group name.

The cube name is case-sensitive; for example, you must type uNorth not unorth, for the example below to be valid. No spaces can appear between the option and its argument.

The basic syntax for using the -u option is as follows:

```
cogtr -u[powercube_name]|[
powercube_group_name/child_cube_name] -p
filename.py?|-mfilename.mdl
```

The *powercube_name* parameter represents the name of the cube in the Transformer model. If *powercube_name* specifies a cube group that is not a time-based partitioned cube, all child cubes of the cube group are copied to the deployment location and activated. If *powercube_name* specifies a time-based partitioned cube, it is processed as a single cube; member cubes are not deployed individually.

If a name conflict exists with a child cube of a cube group that has the same name as another cube in the model, use the *powercube_group_name/child_cube_name* parameters.

This example opens the model file Roofing.mdl, discarding a checkpoint file if one exists. It then creates all defined cubes and writes the partition information of cubes North and East to the log file F:\Roof.log.

```
cogtr -i -uNorth -uEast -mRoofing.mdl -dLogFileName=Roof.log
```

-v option

This option specifies how many data source records to use to create a test cube.

If you have a large data source file, this option enables you to do a test run on a limited number of records before processing the entire file.

If the number of records you specify for the test is greater than the total number of records in the source file, the process runs normally, using the entire file.

Because the option syntax does not reference a model file, -v is used in combination with -c, -m, or -p, each of which does reference a model file.

The basic syntax for using the -v option is as follows:

```
cogtr -vdata_subset_number
```

The following example processes a subset of the records in the binary model file Xyzsales.pyj (525 records), generating categories, and creating the test cube.

```
cogtr -c -pXyzsales.pyj -v525
```

-x option

This option updates the column and measure scales of the MDL model, provided the data source can handle queries about scale. Therefore the option is supported for relational data sources, but not ASCII or other flat-file data sources.

All column scales are first checked to confirm that they match those in the source. Then, the associated measures are checked and their output scales are updated, as required.

The basic syntax for using the -x option is as follows:

```
cogtr -x -mfilename.mdl
```

The following example opens the Field3.mdl file and grants access to the required database using the signon named field, user ID CarlosR, and password pw462. Assuming that the signon matches what is defined in the model, the process runs in batch mode, updates scales of columns defined in the model, and saves it back to the model.

```
cogtr -x -o -s -kfield=CarlosR/pw462 -mField3.mdl
```

-y options

This option specifies how IBM Cognos Series 7 user-class security conversion is performed.

You can use the -y options alone to save changes to the model file , or you can use it in combination with other options.

The -y options correspond to the three security import options in the **Import model with IBM Cognos Series 7 user class view** dialog box. For more information, see the topic, "Upgrade an IBM Cognos Series 7 Secured PowerCube", in the *IBM Cognos Transformer User Guide*.

-y1 option

Choose this option when you want to maintain the view operations applied in the user class views and use the IBM Cognos Series 7 user classes.

This option requires that you configure the IBM Cognos Series 7 security on which the upgraded model was designed as an available namespace in IBM Cognos BI environment. The unique identifier that locates the user class in Access Manager is converted to an IBM Cognos identifier, and this process will fail if you use this option with a different Series 7 namespace.

The following example runs cogtr.exe in batch mode, does not generate cubes or update categories, saves the .mdl file, preserves the IBM Cognos Series 7 user class

views and user classes associated with the IBM Cognos Series 7 model, and logs onto the GOnamespace with the Administrator user name and no password.

```
cogtr.exe -o -s -y1GOnamespace=Administrator/ -mNationalV7.mdl
```

-y2 option

This option imports only the IBM Cognos Series 7 user class views associated with the upgraded model.

Choose this option when you want to import the IBM Cognos Series 7 user class views associated with the model, but not the user classes.

This option allows you to maintain the view operations applied in the IBM Cognos Series 7 user class views but not use a Series 7 namespace with the custom views, or if you do not intend to expose IBM Cognos Series 7 as an a available namespace configured in IBM Cognos BI environment.

The following example runs cogtr.exe in batch mode, does not generate cubes or update categories, saves the .mdl file, and preserves the IBM Cognos Series 7 user class views associated with the IBM Cognos Series 7 model.

```
cogtr.exe -o -s -y2 -mNationalV7.mdl
```

-y3 option

This option discards the IBM Cognos Series 7 user class views and user classes associated with the upgraded model.

Choose this option when you plan to create new custom views and use only the security objects currently configured in the IBM Cognos namespaces.

The following example runs cogtr.exe in batch mode, does not generate cubes or update categories, saves the .mdl file, and discards the IBM Cognos Series 7 user class views and user classes associated with the upgraded IBM Cognos Series 7 model.

```
cogtr.exe -o -s -y3 -mNationalV7.mdl
```

Backward Compatibility

Transformer supports previous versions of preference files.

Sample Preference Files for IBM Cognos Series 7

The command line preference files are the same as those for previous versions of IBM Cognos Transformer. The following sample is an example of a transformer.rc file.

```
LogFileDirectory=./../logs
ModelSaveDirectory=/tmp
DataSourceDirectory=/tmp
CubeSaveDirectory=/tmp
DataWorkDirectory=/tmp
ModelWorkDirectory=/tmp
MaxTransactionNum=500000
LogDetailLevel=0
UseTransDAPipe=0
LogFileName=
LogFileAppend=FALSE
LoggingFrequency=-1
WindowsDateFormat=0
MdcDebugOn=
DatDebugOn=0
```

```
WorkCountOn=0
DumpCSVPath=
ChildRatioThreshold=35
DetachDataSource=TRUE
FilenameVariables=FALSE
IncUpdateWarnings=TRUE
LunarFiscalLabeling=FALSE
OrderByCategoryLabeling=FALSE
ServerVerbOutput=1
DefaultSeparator=,
ThousandSeparator=,
DecimalPoint=.
ServerWaitTimeout=10
ServerWaitPeriods=30
ServerAnimateTimeOut=3
ServerSyncTimeOut=-1
PowerGridBlockSize=16384U
WorkFileCompress=0
PartitionSizeOverride=0
AutoPartitionOff=0
WorkFileMaxSize=1500000000
WorkFileSortSize=8000000
EnablePCOptimizer=TRUE
TransdaPath=
TransdabPath=
CenturyBreak=20
KeepDataFiles=1
LoaderInterval=
LoaderTimeOut=
LoaderFileSize=
MultiFileCubeThreshold=0
HaltOnSecurityError=FALSE
```

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at " Copyright and trademark information " at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

S

T

U

X