

Sentaurus™ Data Explorer User Guide

Version N-2017.09, September 2017

SYNOPSYS®

Copyright and Proprietary Information Notice

©2017 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>. All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

About This Guide	ix
Related Publications	ix
Conventions	ix
Customer Support	ix
Accessing SolvNet	x
Contacting Synopsys Support	x
Contacting Your Local TCAD Support Team Directly	x
<hr/>	
Chapter 1 Using Sentaurus Data Explorer	1
Functionality of Sentaurus Data Explorer	1
File Formats Supported	2
<hr/>	
Chapter 2 Command-Line Interface	3
Using the Command Line	3
Available Commands	4
Parameters and Options	5
Command-Line Help	6
Converting File Formats	8
Syntax	8
Converting TIF to TDR Mixed Element	9
Converting TIF to DF-ISE Grid and Data	9
Converting TDF to TDR Mixed Element	10
Converting TDF to DF-ISE Grid and Data	11
Converting TDR Mixed Element to TIF	11
Converting TDR File to DF-ISE Files	12
Converting DF-ISE Boundary to TDR Boundary	13
Converting DF-ISE Grid and Data to TDR Mixed Element	13
Converting DF-ISE Plot to TDR XY	14
Converting DF-ISE Grid and Data to TIF	15
Converting IVL to TDR XY	15
Converting PLX to TDR XY	16
Mirror Commands	16
Mirroring DF-ISE to DF-ISE	17
Mirroring TDR to TDR	17

Chapter 3 Tcl Interface	19
Overview of Tcl Interface	19
Using the Tcl Interface	20
Accessing Command-Line Arguments From Tcl Script	20
Deleting Items in a Loop	21
Example Scripts	21
Example 1: Extracting Header Information From a File	21
Example 2: Modifying Data Values	22
Example 3: Printing Tags	23
Limitations of Tcl Interface	25
File-Related Functions	25
TdrFileClose	26
TdrFileConvert	27
TdrFileGetNumGeometry	28
TdrFileGetTagGroup	29
TdrFileOpen	30
TdrFileSave	31
Geometry-Related Functions	32
TdrGeometryDelete	33
TdrGeometryGetDimension	34
TdrGeometryGetName	35
TdrGeometryGetNumRegion	36
TdrGeometryGetNumState	37
TdrGeometryGetShift	38
TdrGeometryGetTagGroup	39
TdrGeometryGetTransform	40
TdrGeometryGetType	41
TdrGeometrySetName	42
TdrGeometrySetShift	43
TdrGeometrySetTransform	44
State-Related Functions	45
TdrStateDelete	46
TdrStateGetName	47
TdrStateGetTagGroup	48
TdrStateSetName	49
Region-Related Functions	50
TdrRegionGetDimension	51
TdrRegionGetMaterial	52
TdrRegionGetName	53
TdrRegionGetNumDataset	54

TdrRegionGetTagGroup	55
TdrRegionGetType	56
TdrRegionSetMaterial	57
TdrRegionSetName	58
Dataset-Related Functions	59
TdrDatasetDelete	60
TdrDatasetDeleteByName	61
TdrDatasetGetLocation	62
TdrDatasetGetName	63
TdrDatasetGetNumValue	64
TdrDatasetGetQuantity	65
TdrDatasetGetStructure	66
TdrDatasetGetTagGroup	67
TdrDatasetGetType	68
TdrDatasetGetUnit, TdrDatasetGetUnitLong	69
TdrDatasetRename	70
TdrDatasetRenameQuantity	71
TdrDatasetSetName	72
TdrDatasetSetQuantity	73
Data Value-Related Functions	74
TdrDataGetAllCoordinates	75
TdrDataGetComponent	76
TdrDataGetCoordinate	78
TdrDataGetNumCol	79
TdrDataGetNumRow	80
TdrDataGetValue	81
TdrDataSetComponent	82
Tag Group-Related Functions	83
TdrTagGroupCreate	84
TdrTagGroupDelete	85
TdrTagGroupDeleteByName	86
TdrTagGroupGetByPath	87
TdrTagGroupGetName	89
TdrTagGroupGetNumTag	90
TdrTagGroupGetNumTagGroup	91
TdrTagGroupGetTagGroup	92
Tag-Related Functions	93
TdrTagCreateScalar	94
TdrTagDelete	95
TdrTagDeleteByName	96
TdrTagGetComponent	97

Contents

TdrTagGetName	98
TdrTagGetNumCol	99
TdrTagGetNumRow	100
TdrTagGetStructure	101
TdrTagGetType	102
TdrTagGetValue	103
TdrTagSetComponent	104

Chapter 4 Reference Guide	105
Environment Variables and Configuration Files	105
Supported Conversions	105
TDF-to-TDR Conversions	106
TDF Format Constraints	106
Material Names	106
Quantity Names	106
Conversion Factor	107
Ignoring Unknown Quantities	107
Electrodes and Thermodes	107
Volume Regions With Material Electrode or Thermode	107
Removing Ambient Regions	108
Interface Regions	108
Inconsistent Faces	108
Splitting Rectangles	109
Extracting Boundaries	110
TIF-to-TDR Conversions	110
Material and Quantity Names	110
Removing Contact Regions	110
Missing Ambient Regions	110
TDR-to-TIF Conversions	111
Material and Quantity Names	111
Contacts	111
Interface Regions	111
Region Names	111
Mirroring	111
Number of Regions	111
Naming Regions	112
Vector Datasets	112

Appendix A Tcl Commands Interface of Sentaurus Data Explorer	113
File (TDR Collection) Commands	113
Geometry Commands	113
State Commands	114
Region Commands	114
Dataset Commands	115
Data Value Commands	116
Tag Group Commands	116
Tag Commands	117

Appendix B Structure of TDR Files	119
Overview	119
Tag Groups and Tags	119

Contents

About This Guide

This guide describes the operation of Synopsys Sentaurus™ Data Explorer, which can explore and edit the data produced as output files from simulation processes. With Sentaurus Data Explorer, you can convert these files to the TDR file format, and can view and edit these files.

Related Publications

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNet® support site (see [Accessing SolvNet on page x](#)).
 - Documentation available on SolvNet at <https://solvnet.synopsys.com/DocsOnWeb>.
-

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Blue text	Identifies a cross-reference (only on the screen).
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
<code>Courier font</code>	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
<i>Italicized text</i>	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNet customer support website and by contacting the Synopsys support center.

Accessing SolvNet

The SolvNet support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNet site:

1. Go to the web page at <https://solvnet.synopsys.com>.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

If you need help using the site, click **Help** on the menu bar.

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys [Global Support Centers](#) site on synopsys.com. There you can find e-mail addresses and telephone numbers for Synopsys support centers throughout the world.
 - Go to either the Synopsys SolvNet site or the Synopsys Global Support Centers site and [open a case online](#) (Synopsys user name and password required).
-

Contacting Your Local TCAD Support Team Directly

Send an e-mail message to:

- support-tcad-us@synopsys.com from within North America and South America.
- support-tcad-eu@synopsys.com from within Europe.
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia).
- support-tcad-kr@synopsys.com from Korea.
- support-tcad-jp@synopsys.com from Japan.

This chapter presents an introduction to Sentaurus Data Explorer.

Functionality of Sentaurus Data Explorer

Sentaurus Data Explorer is a tool for editing and converting TDR files.

The TDR file format is the standard format for exchanging data between TCAD Sentaurus tools. Since the TDR format is a binary format and cannot be edited using a text editor, Sentaurus Data Explorer is provided to display, access, and modify the data contained in TDR files. For compatibility with other applications and tools, it is sometimes necessary to have the capability to convert to and from other file formats. Sentaurus Data Explorer allows you to convert formats as required.

Sentaurus Data Explorer has two distinct and different modes of operation: the command-line interface and the Tcl interface.

With the command-line interface, you can apply simple commands to convert files to different formats, and to create new files by copying and modifying files. This mode is most convenient when converting many files at the same time, using a batch file or script, and then continuing to work with these files in other tools.

The Tcl interface allows you to use the full flexibility of Tcl for writing and using scripts.

The main features of Sentaurus Data Explorer include:

- Command-line options to convert files to different formats and to create symmetric structures.
- Tcl interface for TDR for writing and running scripts.
- Reading files in DF-ISE, IVL, PLX, TDF, TDR, and TIF formats.
- Writing files in DF-ISE, TDR, and TIF formats.

File Formats Supported

Sentaurus Data Explorer converts different file formats. For all conversions, the internal representation uses the TDR format. The possible input file formats are:

- DF-ISE (.grd, .dat, .bnd, .plt)
- IVL
- PLX
- TDF
- TDR
- TIF

The possible output file formats are:

- DF-ISE (.grd, .dat, .bnd, .plt)
- TDR
- TIF

For detailed information about supported file formats, see [Converting File Formats on page 8](#) and [Supported Conversions on page 105](#).

NOTE Sentaurus Data Explorer provides functionality for converting data between different file formats. However, it does *not* support extraction of DF-ISE boundaries from DF-ISE grids and TDR mixed-element grids. This extraction is provided by Sentaurus Mesh.

This chapter describes the command-line interface available in Sentaurus Data Explorer.

The command-line interface provides all the documented conversions between the supported data formats and allows you to run Tcl scripts to modify TDR files and to convert them. In addition, it is used to print general information about a TDR file.

Using the Command Line

To use the command-line interface of Sentaurus Data Explorer, you must specify exactly one of the commands together with its arguments on the command line:

```
tdx -short_command [options] [arguments]
```

or:

```
tdx --long_command [options] [arguments]
```

The available commands and options are described here. Each command or option has a short and a long form, which start with one and two dashes, respectively. Writing commands with the short form is quick and easy for direct input on the command line.

In contrast, the long form of the commands is self evident. This form should always be used when writing scripts and batch files.

Available Commands

Table 1 lists the short and long forms of all commands that are available for the command-line interface.

Table 1 Short and long forms of commands

Short form	Long form	Description
-d	--dfise2tdr	Converts DF-ISE grid file only, or a grid file with one, two, or three data files, or a boundary file only, or a plot file only to TDR format.
-dd	--tdr2dfise	Converts TDR file to DF-ISE files.
-df	--dfise2tif	Converts DF-ISE grid and data files to TIF file.
-f	--tif2tdr	Converts TIF file to TDR mixed-element file.
-fd	--tif2dfise	Converts TIF file to DF-ISE grid and data files.
-i	--ivl2tdr	Converts IVL file to TDR xy file.
-mdd	--mirr-dfise	Mirrors the geometry of a DF-ISE file and saves the result to another DF-ISE file.
-mtt	--mirr-tdr	Mirrors TDR geometry and saves the result to another TDR file.
-p	--plx2tdr	Converts PLX file to TDR xy plot file.
-t	--tdf2tdr	Converts TDF file to TDR mixed-element file.
-td	--tdf2dfise	Converts TDF file to DF-ISE grid and data files.
-tf	--tdr2tif	Converts TDR mixed element to TIF file.
-ts	--tdr-change-cs	Converts the <i>traditional</i> (DF-ISE) coordinate system to the Sentaurus Process coordinate system, and vice versa.

Parameters and Options

Table 2 lists all available options for commands with their type, default value, and description.

Table 2 Command options for command-line interface

Parameter/Option		Type	Default	Description
Short form	Long form			
-a	--ignore-ambient-regions	Boolean	false	Do not convert regions for which the material is ambient.
-c	--ignore-conductor-regions	Boolean	false	Do not convert regions for which the material or parent material is conductor.
-m	--geometry-name	String	""	TDR geometry name.
-M	--geometry-index	Integer	-1	TDR geometry index.
-q	--ignore-nondatex-quantities	Boolean	false	Do not convert fields for which there is no DATEX quantity name in the sol.db file.
-r	--split-rectangles	Boolean	false	For a 2D geometry, split rectangles into triangles.
-ren	--rename	String	""	Rename a region or regions.
-s	--state-name	String	""	TDR state name.
-S	--state-index	Integer	-1	TDR state index.
-sp	--sprocess	Boolean	false	Convert to Sentaurus Process coordinate system.
-tr	--traditional	Boolean	false	Convert to traditional (DF-ISE) coordinate system.
-w	--do-not-swap-3d-coord	Boolean	false	Do not swap 3D coordinates.
-x	--xmin	Boolean	false	Mirror at xmin.
-X	--xmax	Boolean	false	Mirror at xmax.
-xy	--xy-name	String	""	TDR xy plot name.
-XY	--xy-index	Integer	-1	TDR xy plot index.
-y	--ymin	Boolean	false	Mirror at ymin.
-Y	--ymax	Boolean	false	Mirror at ymax.
-z	--zmin	Boolean	false	Mirror at zmin.
-Z	--zmax	Boolean	false	Mirror at zmax.

2: Command-Line Interface

Command-Line Help

Command-Line Help

When the command-line option `-h` or `--help` is used, the following text is displayed, which shows a summary of the different commands, their options, and their arguments:

Batch mode (-/-- command)	Parameter (-param)	Source (*Base Name)	Destination (*Base Name) [] Optional	Description
Convert:				
<code>fd</code>	<code>tif2dfise</code>	<code>a,c,q,r</code>	<code>*<TIF></code>	<code>[*<DF-ISE>]</code> TIF to DF-ISE file
<code>f</code>	<code>tif2tdr</code>	<code>a,c,q,r</code>	<code>*<TIF></code>	<code>[*<TDR>]</code> TIF to TDR file
<code>td</code>	<code>tdf2dfise</code>	<code>a,c,q,r,w</code>	<code>*<TDF></code>	<code>[*<DF-ISE>]</code> TDF to DF-ISE file
<code>t</code>	<code>tdf2tdr</code>	<code>a,c,q,r,w</code>	<code>*<TDF></code>	<code>[*<TDR>]</code> TDF to TDR file
<code>tf</code>	<code>tdr2tif</code>	<code>m,M,s,S</code>	<code>*<TDR></code>	<code>[*<TIF>]</code> TDR to TIF file
<code>dd</code>	<code>tdr2dfise</code>	<code>m,M,s,S</code>	<code>*<TDR></code>	<code>[*<GRD>]</code> TDR to DF-ISE file
<code>df</code>	<code>dfise2tif</code>		<code><DF-ISE></code>	<code>[*<TIF>]</code> DF-ISE to TIF file
<code>d</code>	<code>dfise2tdr</code>		<code><DF-ISE></code>	<code>[*<TDR>]</code> DF-ISE to TDR file
<code>d</code>	<code>dfise2tdr</code>		<code><GRD></code>	<code>[*<TDR>]</code> with Gridfile and 1 to 3 Datfiles
<code>d</code>	<code>dfise2tdr</code>		<code>[<DAT>]</code>	
<code>d</code>	<code>dfise2tdr</code>		<code><BND></code>	<code>[*<TDR>]</code> or Boundaryfile
<code>d</code>	<code>dfise2tdr</code>		<code><PLT></code>	<code>[*<TDR>]</code> or Plotfile
<code>i</code>	<code>ivl2tdr</code>		<code>*<IVL></code>	<code>[*<TDR>]</code> IVL to TDR file
<code>p</code>	<code>plx2tdr</code>		<code>*<PLX></code>	<code>[*<TDR>]</code> PLX to TDR file
<code>ts</code>	<code>tdr-change-cs</code>	<code>tr,sp</code>	<code>*<TDR></code>	<code>[*<TDR>]</code> TDR to TDR with another coordinate system
Mirror:				
<code>mtt</code>	<code>mirr-tdr</code>		<code>*<TDR></code>	<code>*<TDR></code> Mirror TDR to TDR
<code>mdd</code>	<code>mirr-dfise</code>		<code>*<DF-ISE></code>	<code>*<DF-ISE></code> DF-ISE to DF-ISE
		<code>m,M,s,S</code>		Mirror name,index
		<code>x,X,y,Y,z,Z</code>		Mirror at x,y,z
		<code>ren reg=new/...</code>		Rename region(s) reg to new name
<code>-tcl script_name</code> runs the specified Tcl script file.				
<code>-tclcmd <tcl command with parameters></code> runs a single Tcl command.				
<code>-h</code> or <code>--help</code> prints this help				


```
-v or --version  
    prints the version of TDX and checks the availability of its license  
  
-info tdr_file  
    prints general information about TDR file
```

The table shows conversion options. The first and second columns contain the short and long names of all available commands, respectively. The third column lists the parameters of the commands. Optional parameters are enclosed in brackets. The source and destination columns indicate the file format of the input and output files, respectively.

For many commands, the destination is optional. If the destination is not specified, the name of the output file is constructed from the base name of the input file and the extension appropriate for the type of output file. The base name consists of all characters in a file name up to (but not including) the last '.' character. Using the base name is possible for all entries in the source and destination columns marked with an asterisk.

After the conversion options table, there is the list of other command-line options. The `-info` option prints general information about the TDR file. The following data is displayed:

- File name and number of geometries
- For each geometry:
 - Geometry name, type, and dimension
 - Transformation matrix and shift vector
 - Numbers of states, vertices, edges, faces, elements, and material elements
- For each region:
 - Region index, name, material or other property, such as “Contact”
 - Number of datasets, number of elements by type

Converting File Formats

The following sections provide a detailed description of all conversions available using the command-line interface. The conversions are presented in the following order:

- TIF to TDR and DF-ISE
- TDF to TDR and DF-ISE
- TDR to TIF and DF-ISE
- DF-ISE to TDR and TIF
- IVL to TDR
- PLX to TDR

For detailed information about the effects of different conversion options, see [Supported Conversions on page 105](#) to [TDR-to-TIF Conversions on page 111](#).

Syntax

The following special characters are used in the syntax descriptions: angle brackets <>, brackets [], parentheses (), and vertical bar |. These characters are only used in the syntax description and are not part of the actual input.

A lowercase letter in angle brackets represents a value of a given type that must be substituted by the user:

<n> numeric value
<s> string value

For example:

```
--geometry-name <s> --state-index <n>
```

indicates that a string value must be specified following the command-line option `--geometry-name` and a numeric value must be specified following the option `--state-index`.

Brackets enclose optional command-line arguments and parameters.

Parentheses are used to indicate the grouping of command-line options and their arguments.

The vertical bar is used to separate entries in a list from which exactly one entry must be specified.

In the following sections, the syntax of each command is described twice. First using only the short form and then using only the long form. Of course, it is possible to use a combination of long and short forms.

Converting TIF to TDR Mixed Element

Syntax

```
tdx -f [-a] [-c] [-q] [-r] tif_source_base_name [tdr_destination_base_name]

tdx --tif2tdr [--ignore-ambient-regions] [--ignore-conductor-regions] \
  [--ignore-nondatex-quantities] [--split-rectangles] tif_source_base_name \
  [tdr_destination_base_name]
```

Examples

1. `tdx -f tif_file.tif tdr_file`
Input: `tif_file.tif`
Output: `tdr_file.tdr`
2. `tdx --tif2tdr tif_file.tif tdr_file`
Input: `tif_file.tif`
Output: `tdr_file.tdr`
3. `tdx --tif2tdr tif_file`
Input: `tif_file.tif`
Output: `tif_file.tdr`

Converting TIF to DF-ISE Grid and Data

Syntax

```
tdx -fd [-a] [-c] [-q] [-r] tif_source_base_name [dfise_destination_base_name]

tdx --tif2dfise [--ignore-ambient-regions] [--ignore-conductor-regions] \
  [--ignore-nondatex-quantities] [--split-rectangles] tif_source_base_name \
  [dfise_destination_base_name]
```

Examples

1. `tdx -fd tif_file.tif dfise_file`
Input: `tif_file.tif`
Output: `dfise_file.grd`, `dfise_file.dat`

2: Command-Line Interface

Converting File Formats

2. `tdx --tif2dfise tif_file.tif dfise_file`

Input: `tif_file.tif`

Output: `dfise_file.grd, dfise_file.dat`

3. `tdx --tif2dfise tif_file`

Input: `tif_file.tif`

Output: `tif_file.grd, tif_file.dat`

Converting TDF to TDR Mixed Element

Syntax

```
tdx -t [-a] [-c] [-q] [-r] [-w] tdf_source_base_name \  
      [tdr_destination_base_name]
```

```
tdx --tdf2tdr [--ignore-ambient-regions] [--ignore-conductor-regions] \  
      [--ignore-nondatex-quantities] [--split-rectangles] \  
      [--do-not-swap-3d-coord] tdf_source_base_name [tdr_destination_base_name]
```

Examples

1. `tdx -t tdf_file.tdf tdr_file`

Input: `tdf_file.tdf`

Output: `tdr_file.tdr`

2. `tdx --tdf2tdr tdf_file.tdf tdr_file`

Input: `tdf_file.tdf`

Output: `tdr_file.tdr`

3. `tdx --tdf2tdr tdf_file`

Input: `tdf_file.tdf`

Output: `tdf_file.tdr`

Converting TDF to DF–ISE Grid and Data

Syntax

```
tdx -td [-a] [-c] [-q] [-r] [-w] tdf_source_base_name \  
    [dfise_destination_base_name]  
  
tdx --tdf2dfise [--ignore-ambient-regions] [--ignore-conductor-regions] \  
    [--ignore-nondatex-quantities] [--split-rectangles]  
    [--do-not-swap-3d-coord] tdf_source_base_name \  
    [dfise_destination_base_name]
```

Examples

1. `tdx -td tdf_file.tdf dfise_file`
Input: `tdf_file.tdf`
Output: `dfise_file.grd`, `dfise_file.dat`
2. `tdx --tdf2dfise tdf_file.tdf dfise_file`
Input: `tdf_file.tdf`
Output: `dfise_file.grd`, `dfise_file.dat`
3. `tdx --tdf2dfise tdf_file`
Input: `tdf_file.tdf`
Output: `tdf_file.grd`, `tdf_file.dat`

Converting TDR Mixed Element to TIF

Syntax

```
tdx -tf (-m <s>)|(-M <n>) [(-s <s>)|(-S <n>)] tdr_source_base_name \  
    [tif_destination_base_name]  
  
tdx --tdr2tif (--geometry-name <s>)|(--geometry-index <n>) \  
    [(-state-name <s>)|(--state-index <n>)] tdr_source_base_name \  
    [tif_destination_base_name]
```

Examples

1. `tdx -tf -M 0 tdr_file.tdr tif_file`
Input: `tdr_file.tdr`
Output: `tif_file.tif`

2: Command-Line Interface

Converting File Formats

2. `tdx --tdr2tif -M 0 tdr_file.tdr tif_file`

Input: `tdr_file.tdr`

Output: `tif_file.tif`

3. `tdx --tdr2tif -M 0 tdr_file`

Input: `tdr_file.tdr`

Output: `tdr_file.tif`

Converting TDR File to DF-ISE Files

Syntax

```
tdx -dd (-m <s>)|(-M <n>) [(-s <s>)|(-S <n>)] tdr_source_base_name \  
[dfise-destination_base_name]
```

```
tdx --tdr2dfise (--geometry-name <s>)|(--geometry-index <n>) \  
[(--state-name <s>)|(--state-index <n>)] tdr_source_base_name \  
[dfise-destination_base_name]
```

Examples

1. `tdx -dd -M 0 -S 0 tdr_file.tdr dfise_file`

Input: `tdr_file.tdr`

Output: `dfise_file.grd, dfise_file.dat` using geometry with index 0 and state with index 0

2. `tdx --tdr2dfise -M 0 -S 0 tdr_file.tdr dfise_file`

Input: `tdr_file.tdr`

Output: `dfise_file.grd, dfise_file.dat` using geometry with index 0 and state with index 0

3. `tdx --tdr2dfise -m geometry_0 -s state_0 tdr_file.tdr dfise_file`

Input: `tdr_file.tdr`

Output: `dfise_file.grd, dfise_file.dat` using geometry with name `geometry_0` and state with name `state_0`

Converting DF–ISE Boundary to TDR Boundary

Syntax

```
tdx -d dfise_source_bnd [tdr_destination_base_name]
tdx --dfise2tdr dfise_source_bnd [tdr_destination_base_name]
```

Examples

1. `tdx -d dfise_file.bnd tdr_file`
Input: `dfise_file.bnd`
Output: `tdr_file.tdr`
2. `tdx --dfise2tdr dfise_file.bnd tdr_file`
Input: `dfise_file.bnd`
Output: `tdr_file.tdr`

Converting DF–ISE Grid and Data to TDR Mixed Element

Syntax

```
tdx -d dfise_source_grd [dfise_source1_dat [dfise_source2_dat \
    [dfise_source3_dat]]] [tdr_destination_base_name]
tdx -d dfise_source_bnd [tdr_destination_base_name]
tdx -d dfise_source_plt [tdr_destination_base_name]
tdx --dfise2tdr dfise_source_grd [dfise_source1_dat [dfise_source2_dat \
    [dfise_source3_dat]]] [tdr_destination_base_name]
tdx --dfise2tdr dfise_source_bnd [tdr_destination_base_name]
tdx --dfise2tdr dfise_source_plt [tdr_destination_base_name]
```

Examples

1. `tdx -d dfise_file.grd tdr_file`
Input: `dfise_file.grd`
Output: `tdr_file.tdr`
2. `tdx --dfise2tdr dfise_file.grd tdr_file`
Input: `dfise_file.grd`
Output: `tdr_file.tdr`

2: Command-Line Interface

Converting File Formats

3. `tdx --dfise2tdr dfise_file.grd dfise_file.dat tdr_file`
Input: `dfise_file.grd, dfise_file.dat`
Output: `tdr_file.tdr`
4. `tdx --dfise2tdr dfise_file.grd dfise_dat_file_1.dat \`
`dfise_dat_file_2.dat tdr_file`
Input: `dfise_file.grd, dfise_dat_file_1.dat dfise_dat_file_2.dat`
Output: `tdr_file.tdr`
5. `tdx --dfise2tdr dfise_file.plt tdr_file`
Input: `dfise_file.plt`
Output: `tdr_file.tdr`

Converting DF-ISE Plot to TDR XY

Syntax

```
tdx -d dfise_source_plt [tdr_destination_base_name]
tdx --dfise2tdr dfise_source_plt [tdr_destination_base_name]
```

Examples

1. `tdx -d dfise_file.plt tdr_file`
Input: `dfise_file.plt`
Output: `tdr_file.tdr`
2. `tdx --dfise2tdr dfise_file.plt tdr_file`
Input: `dfise_file.plt`
Output: `tdr_file.tdr`

Converting DF–ISE Grid and Data to TIF

Syntax

```
tdx -df dfise_source_grd [dfise_source_dat] [tif_destination_base_name]
tdx --dfise2tif dfise_source_grd [dfise_source_dat] \
  [tif_destination_base_name]
```

Examples

1. `tdx -df dfise_file.grd dfise_file.dat tif_file`
Input: `dfise_file.grd`, `dfise_file.dat`
Output: `tif_file.tif`
2. `tdx --dfise2tif dfise_file.grd dfise_file.dat tif_file`
Input: `dfise_file.grd`, `dfise_file.dat`
Output: `tif_file.tif`

Converting IVL to TDR XY

Syntax

```
tdx -i ivl_source_base_name [tdr_destination_base_name]
tdx --ivl2tdr ivl_source_base_name [tdr_destination_base_name]
```

Examples

1. `tdx -i ivl_file tdr_file`
Input: `ivl_file.ivl`
Output: `tdr_file.tdr`
2. `tdx --ivl2tdr ivl_file tdr_file`
Input: `ivl_file.ivl`
Output: `tdr_file.tdr`
3. `tdx --ivl2tdr ivl_file`
Input: `ivl_file.ivl`
Output: `ivl_file.tdr`

Converting PLX to TDR XY

Syntax

```
tdx -p plx_source_base_name [tdr_destination_base_name]
```

```
tdx --plx2tdr plx_source_base_name [tdr_destination_base_name]
```

Examples

1. `tdx -p plx_file tdr_file`

Input: `plx_file.ivl`

Output: `tdr_file.tdr`

2. `tdx --plx2tdr plx_file tdr_file`

Input: `plx_file.ivl`

Output: `tdr_file.tdr`

3. `tdx --plx2tdr plx_file`

Input: `plx_file.ivl`

Output: `plx_file.tdr`

Mirror Commands

The mirror commands create a symmetric geometry by reflecting the input geometry with respect to a mirror axis (point in 1D, line in 2D, plane in 3D). In 2D and 3D, the mirror axis is always perpendicular to one of the coordinate axes. The location of the mirror axis can be chosen to be the minimum or maximum coordinate of the input geometry in the direction perpendicular to the mirror axis.

Table 3 Mirror options

Option	Mirror axis perpendicular to	Located at
-x	x-axis	Minimum x-coordinate
-X	x-axis	Maximum x-coordinate
-y	y-axis	Minimum y-coordinate
-Y	y-axis	Maximum y-coordinate
-z	z-axis	Minimum z-coordinate
-Z	z-axis	Maximum z-coordinate

By default, the name of the mirrored region is the name of the original region with the suffix `_mirrored`. Using the option `-ren`, it is possible to specify new names for the mirrored regions.

Mirroring DF-ISE to DF-ISE

Syntax

```
tdx -mdd -x|-X|-y|-Y|-z|-Z [-ren orig_reg_name_1=new_reg_name_1[/...]] \  
    dfise_source_base_name dfise_destination_base_name  
  
tdx --mirr-dfise --xmin|--xmax|--ymin|--ymax|--zmin|--zmax \  
    [--rename orig_reg_name_1=new_reg_name_1[/...]] dfise_source_base_name \  
    dfise_destination_base_name
```

Examples

1. `tdx -mdd -y dfise_file dfise_mirr`
Input: `dfise_file.grd, dfise_file.dat`
Output: `dfise_mirr.grd, dfise_mirr.dat`
2. `tdx --mirr-dfise -y dfise_file dfise_mirr`
Input: `dfise_file.grd, dfise_file.dat`
Output: `dfise_mirr.grd, dfise_mirr.dat`
3. `tdx --mirr-dfise -y -ren silicon=silicon_mir dfise_file dfise_mirr`
Input: `dfise_file.grd, dfise_file.dat`
Output: `dfise_mirr.grd, dfise_mirr.dat`

The region with the default name `silicon_mirrored` will be renamed `silicon_mir`.

Mirroring TDR to TDR

Syntax

```
tdx -mtt -x|-X|-y|-Y|-z|-Z [-ren orig_reg_name_1=new_reg_name_1[/...]] \  
    tdr_source_base_name tdr_destination_base_name  
  
tdx --mirr-tdr --xmin|--xmax|--ymin|--ymax|--zmin|--zmax \  
    [--rename orig_reg_name_1=new_reg_name_1[/...]] tdr_source_base_name \  
    tdr_destination_base_name
```

2: Command-Line Interface

Mirror Commands

Examples

1. `tdx -mtt -y tdr_file.tdr tdr_dfise_mirr`

Input: `tdr_file.tdr`

Output: `tdr_dfise_mirr.grd, tdr_dfise_mirr.dat`

2. `tdx --mirr-tdr -y tdr_file.tdr tdr_dfise_mirr`

Input: `tdr_file.tdr`

Output: `tdr_dfise_mirr.grd, tdr_dfise_mirr.dat`

3. `tdx -mtt -y -ren "region_5=mirr region 5/region_1=mirr region 1" \
test2.tdr test2_mirr.tdr`

Input: `test2.tdr`

Output: `test2_mirr.tdr`

The mirrored region `region_5` will be named "mirr region 5", and the mirrored region `region_1` will be named "mirr region 1". If any name contains spaces, the entire name must be enclosed in double quotation marks as in this example.

This chapter presents the Tcl interface available in Sentaurus Data Explorer.

Overview of Tcl Interface

The Tcl interface of Sentaurus Data Explorer is based on the tool command language (Tcl). An input script of Sentaurus Data Explorer is actually a Tcl script and, therefore, enables the full flexibility of Tcl. You may want to use the Tcl interface to extract and process information from a TDR file or to modify certain entries, such as names of materials and datasets. You may also find it useful to add custom information to objects in a TDR file using tags.

You can write and use scripts, giving you the ability to perform tasks more efficiently. The Tcl interface gives you the ability to execute all commands, described in this chapter.

The command syntax is simple and intuitive. The full list of all available Tcl commands for the interface can be found in [Appendix A on page 113](#). Using these commands makes it possible to access and modify data in TDR files easily. However, a proper understanding of the TDR file structure is necessary before you can start writing scripts. A description of the TDR file structure and its parts can be found in [Appendix B on page 119](#).

This chapter provides detailed descriptions of the Tcl interface functions that work with TDR files and plot objects in Sentaurus Data Explorer. The available functions are:

- File-related functions
- Geometry-related functions
- State-related functions
- Region-related functions
- Dataset-related functions
- Data value-related functions
- Tag group-related functions
- Tag-related functions

Most of the functions take integer arguments to specify list entries by index. As a general rule, these indices start from zero, that is, the first entry is referenced by the index 0.

Using the Tcl Interface

To run the Tcl script, use the command:

```
tdx -tcl [Script file]
```

For example:

```
tdx -tcl script.tcl
```

This command runs the Tcl script to save the file `script.tcl`.

In addition, it is possible to run a single Tcl command using the `-tclcmd` option. The syntax of the command is:

```
tdx -tclcmd [tcl command with parameters]
```

For example:

```
tdx -tclcmd TdrFileOpen tdr_file.tdr
```

Accessing Command-Line Arguments From Tcl Script

When writing a general-purpose Tcl script, you may want to access and use command-line arguments, for example, to allow the user of the script to specify a file name.

Sentaurus Data Explorer provides command-line arguments in the Tcl array named `cmd_args`. The following variants are available for convenience:

`$cmd_args(all)`

Contains the complete command-line arguments including the name of the invoking executable file.

`$cmd_args(rest)`

All arguments except `-tcl`, the name of the script file, and the name of the invoking executable file.

`$cmd_args(-tcl)`

Contains the name of the Tcl script file.

Deleting Items in a Loop

The Tcl interface of Sentaurus Data Explorer supports several functions for deleting items by their indices:

- `TdrDatasetDelete`
- `TdrGeometryDelete`
- `TdrStateDelete`
- `TdrTagDelete`
- `TdrTagGroupDelete`

When calling any of these functions in a loop, you must traverse items in *reverse order*, for example:

```
for { set index N-1 } { $index >= 0 } { incr index -1 } {  
    ...  
    TdrStateDelete ... $index ...  
    ...  
}
```

In this example, after deleting a state, the remaining states are re-indexed for structure consistency.

NOTE Traversing items in forward order leads to undefined behavior.

Example Scripts

The following examples are provided to show you how the commands are used together in the context of a script. They also demonstrate how you typically navigate through the content of a TDR file.

Example 1: Extracting Header Information From a File

This script opens a file and lists its geometries. For each geometry, the regions and states are listed:

```
set f myfile.tdr  
puts "file: $f"  
TdrFileOpen $f  
# loop through geometries  
set ng [TdrFileGetNumGeometry $f]
```

3: Tcl Interface

Example Scripts

```
puts "#geometries: $ng"
for {set ig 0} {$ig < $ng} {incr ig} {
    set gname [TdrGeometryGetName $f $ig]
    set ns [TdrGeometryGetNumState $f $ig]
    set nr [TdrGeometryGetNumRegion $f $ig]
    puts "    geometry $ig: $gname"
    puts "        type      : [TdrGeometryGetType $f $ig]"
    puts "        dimension: [TdrGeometryGetDimension $f $ig]"
    puts "        transform: [TdrGeometryGetTransform $f $ig]"
    puts "        shift      : [TdrGeometryGetShift $f $ig]"
    puts "        #states   : $ns"
    # loop through states
    for {set is 0} {$is < $ns} {incr is} {
        set sname [TdrStateGetName $f $ig $is]
        puts "    state $is: $sname"
    }
    puts "        #regions: $nr"
    # loop through regions
    for {set ir 0} {$ir < $nr} {incr ir} {
        set rname [TdrRegionGetName $f $ig $ir]
        puts "    region $ir: $rname"
    }
}
TdrFileClose $f
```

Example 2: Modifying Data Values

This script opens a file, loops through all geometries and their states, and modifies all values of all datasets:

```
set inp original.tdr
set out modified.tdr
TdrFileOpen $inp
# loop through geometries
set ng [TdrFileGetNumGeometry $inp]
for {set ig 0} {$ig < $ng} {incr ig} {
    set ns [TdrGeometryGetNumState $inp $ig]
    set nr [TdrGeometryGetNumRegion $inp $ig]
    # loop through states
    for {set is 0} {$is < $ns} {incr is} {
        # loop through regions
        for {set ir 0} {$ir < $nr} {incr ir} {
            # loop through datasets
            set nd [TdrRegionGetNumDataset $inp $ig $ir $is]
            for {set id 0} {$id < $nd} {incr id} {
                # loop through data values
                set nv [TdrDatasetGetNumValue $inp $ig $ir $is $id]
```



```

for {set iv 0} {$iv < $nv} {incr iv} {
    # loop through components of the data value
    set ni [TdrDataGetNumRow $inp $ig $ir $is $id $iv]
    set nj [TdrDataGetNumCol $inp $ig $ir $is $id $iv]
    for {set i 0} {$i < $ni} {incr i} {
        for {set j 0} {$j < $nj} {incr j} {
            set original [TdrDataGetComponent $inp $ig $ir $is $id $iv $i $j]
            set modified [expr $original + 1]
            TdrDataSetComponent $inp $ig $ir $is $id $iv $i $j $modified
        }
    }
}
}
}
}
}
}
}
}
}
TdrFileSave $inp $out
TdrFileClose $inp

```

Example 3: Printing Tags

This script opens a file and lists the tags and tag groups of all regions and states of all geometries and of the file itself:

```

set f myfile.tdr
set recursive 1

proc PrintTagGroup {tg indent recursive} {
    set space [format "% ${indent}s" ""]
    set nt [TdrTagGroupGetNumTag $tg]
    set ng [TdrTagGroupGetNumTagGroup $tg]
    if {$ng > 0 || $nt > 0} {
        puts "${space}tag group: \'[TdrTagGroupName $tg]\'"
        puts "${space}    contains $ng tag groups and $nt tags"
        # list tags
        for {set it 0} {$it < $nt} {incr it} {
            set struc [TdrTagGetStructure $tg $it]
            set type [TdrTagGetType $tg $it]
            puts "${space}    tag $it:"
            puts "${space}        name: [TdrTagName $tg $it]"
            puts "${space}    structure: $struc"
            puts "${space}        type: $type"
            if {$struc == "scalar"} {
                puts "${space}    value: [TdrTagGetValue $tg $it]"
            } else {
                puts "${space}    rows: [TdrTagGetNumRow $tg $it]"
                puts "${space}    cols: [TdrTagGetNumCol $tg $it]"
            }
        }
    }
}

```

3: Tcl Interface

Example Scripts

```
        puts "${space} value: <not printed out>"
    }
}
# list tag groups
for {set ig 0} {$ig < $ng} {incr ig} {
    set tgi [TdrTagGroupGetTagGroup $tg $ig]
    if {$recursive} {
        PrintTagGroup $tgi [expr $indent + 3] $recursive
    } else {
        puts "${space} tag group $ig: \"[TdrTagGroupGetName $tgi]\""
    }
}
}
}

TdrFileOpen $f
puts "file $f"
PrintTagGroup [TdrFileGetTagGroup $f] 3 $recursive

# loop through geometries
set ng [TdrFileGetNumGeometry $f]
for {set ig 0} {$ig < $ng} {incr ig} {
    set gname [TdrGeometryGetName $f $ig]
    set ns [TdrGeometryGetNumState $f $ig]
    set nr [TdrGeometryGetNumRegion $f $ig]
    puts "    geometry $ig: $gname"
    puts "        type      : [TdrGeometryGetType $f $ig]"
    puts "        dimension: [TdrGeometryGetDimension $f $ig]"
    PrintTagGroup [TdrGeometryGetTagGroup $f $ig] 6 $recursive
    puts "        #regions: $nr"
    # loop through regions
    for {set ir 0} {$ir < $nr} {incr ir} {
        set rname [TdrRegionGetName $f $ig $ir]
        puts "            region $ir: $rname"
        PrintTagGroup [TdrRegionGetTagGroup $f $ig $ir] 9 $recursive
    }
    # loop through states
    puts "        #states: $ns"
    for {set is 0} {$is < $ns} {incr is} {
        set sname [TdrStateGetName $f $ig $is]
        puts "            state $is: $sname"
        PrintTagGroup [TdrStateGetTagGroup $f $ig $is] 9 $recursive
    }
}
}
TdrFileClose $f
```

Limitations of Tcl Interface

Using the Tcl interface, you can read and modify most of the data in a TDR file. Creating new data is not possible except for tags and tag groups. Further limitations are:

- Complex numbers are currently not supported, that is, datasets containing complex values cannot be read or modified.
- Access to coordinates of geometric entities such as vertices, edges, and elements is not provided.

File-Related Functions

[Table 4](#) lists all of the file-related Tcl commands that are available in Sentaurus Data Explorer.

Table 4 File-related functions of Tcl interface for TDR

Command	Description
TdrFileClose	Closes the specified file.
TdrFileConvert	Converts files of different formats. For the syntax of the option <code>convert-command</code> , see Converting File Formats on page 8 .
TdrFileGetNumGeometry	Returns number of geometries.
TdrFileGetTagGroup	Returns handle of tag group.
TdrFileOpen	Opens TDR file. The command must be called before any other function is available when working with the TDR file.
TdrFileSave	Saves a copy of the specified file with a new name or overwrites the saved file.

TdrFileClose

Description

This command closes a TDR file without saving any modifications.

Syntax

```
TdrFileClose <filename>
```

Arguments

Argument	Description
filename	Name of a TDR file.

Return Value

Type of return value is Boolean. It is `TRUE` if a file is closed successfully; otherwise, `FALSE`. For example, it returns `FALSE` if the name is wrong or the file is not opened.

Example

```
TdrFileClose file1.tdr
```

This command closes the file `file1.tdr`.

TdrFileConvert

Description

This command converts a file from one format to another. The syntax of this command is the same as for the corresponding command that is available from the command-line interface (see [Converting File Formats on page 8](#)).

Syntax

```
TdrFileConvert <convert-command> [parameter] <source-file>
    [<destination-file>]
```

Arguments

For the full list of options, see [Converting File Formats on page 8](#).

Argument	Description
convert-command	One of the specified conversion commands such as <code>fd</code> or <code>tif2dfise</code> , which is used to convert a TIF file to a DF-ISE file.
destination-file	Output file name. If the extension of the file is not specified or if it is wrong, the correct extension is appended to the base name of the conversion direction. If the output file already exists, "new" is added before the extension so that the existing file is not overwritten.
parameter	This parameter is not valid for all conversions. It may specify, for example, the type of mirroring (which axis and at min. or max.), or some flags that do not convert regions for which the material is ambient.
source-file	Input file name.

Return Value

Type of return value is Boolean. It is `TRUE` if a conversion was successful; otherwise, `FALSE`.

Example

```
TdrFileConvert -mtt -y tdr_file.tdr tdr_mirr.tdr
```

This example mirrors the file `tdr_file.tdr` at `ymin`. The result is saved to the `tdr_mirr.tdr` file. For the full list of examples, see [Converting File Formats on page 8](#).

TdrFileGetNumGeometry

Description

This command returns the number of geometries in a file.

Syntax

```
TdrFileGetNumGeometry <filename>
```

Arguments

Argument	Description
filename	Name of a TDR file.

Return Value

Type of return value is integer. It shows the number of geometries in a TDR file. It returns a negative value if an error occurs.

Example

```
set file1_num_geom [TdrFileGetNumGeometry file1.tdr]
```

This example sets `file1_num_geom` to the number of geometries in the TDR file named `file1.tdr`.

TdrFileGetTagGroup

Description

This command returns the tag-group handle of a TDR file. This handle can be used in the commands of the tag group–related functions and tag-related functions (see [Tag Group–Related Functions on page 83](#) and [Tag-Related Functions on page 93](#)).

Syntax

```
TdrFileGetTagGroup <filename>
```

Arguments

Argument	Description
filename	Name of a TDR file.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group–related functions and tag-related functions.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`.

TdrFileOpen

Description

This command opens a TDR file.

NOTE This operation is necessary before any other function can be used with the file.

Syntax

```
TdrFileOpen <filename> [-native_units] [-reference_coordinates]
```

Arguments

Argument	Description
filename	Name of a TDR file.
-native_units	If specified, no unit scaling is applied, that is, data is read as written by the tool that wrote the file. Without this option, all data is transformed to standard DATEX units.
-reference_coordinates	If specified, coordinates and vector datasets are transformed into the reference coordinate system.

Return Value

Type of return value is Boolean. It is `TRUE` if a file is opened successfully; otherwise, `FALSE`.

Example

```
TdrFileOpen file1.tdr -native_units
```

This command opens the file `file1.tdr`; data is read in unscaled.

TdrFileSave

Description

This command saves all changes made to a specified file or saves the specified file including all changes with a new name.

Syntax

```
TdrFileSave <filename> [<new_filename>]
```

Arguments

Argument	Description
filename	Name of a TDR file.
new_filename	Optional name for the file, where all changes and data are saved.

Return Value

Type of return value is Boolean. It is TRUE if a file is saved successfully; otherwise, FALSE.

Example

```
TdrFileSave file1.tdr file1_copy.tdr
```

```
TdrFileSave file2.tdr
```

The first example saves a copy of the file `file1.tdr`, including all modifications, to `file1_copy.tdr`.

The second example saves all changes to the same file.

3: Tcl Interface

Geometry-Related Functions

Geometry-Related Functions

Table 5 lists all of the geometry-related Tcl commands that are available in Sentaurus Data Explorer.

Table 5 Geometry-related functions of Tcl interface for TDR

Command	Description
TdrGeometryDelete	Deletes specified geometry.
TdrGeometryGetDimension	Returns dimension of geometry.
TdrGeometryGetName	Returns name of geometry.
TdrGeometryGetNumRegion	Returns number of regions in geometry.
TdrGeometryGetNumState	Returns number of states in geometry.
TdrGeometryGetShift	Returns shifting part of transformation matrix of geometry.
TdrGeometryGetTagGroup	Returns tag-group handle of geometry.
TdrGeometryGetTransform	Returns rotation matrix of geometry.
TdrGeometryGetType	Returns type of geometry.
TdrGeometrySetName	Sets new name for geometry.
TdrGeometrySetShift	Sets new shifting part of transformation matrix of geometry.
TdrGeometrySetTransform	Sets new rotation matrix of geometry.

TdrGeometryDelete

Description

This command deletes a geometry.

NOTE When calling `TdrGeometryDelete` in a loop, you must traverse geometries in reverse order (see [Deleting Items in a Loop on page 21](#)).

Syntax

```
TdrGeometryDelete <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is Boolean. It is `TRUE` if the operation is successful; otherwise, `FALSE`.

Example

```
TdrGeometryDelete file1.tdr 1
```

This example deletes the second geometry of the specified file.

TdrGeometryGetDimension

Description

The command returns the dimension of a geometry.

Syntax

```
TdrGeometryGetDimension <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is integer. It contains the dimension of a specified geometry.

Example

```
set geom_dim [TdrGeometryGetDimension file1.tdr 1]
```

This example sets `geom_dim` to the dimension of the specified geometry.

TdrGeometryGetName

Description

This command returns the name of a geometry.

Syntax

```
TdrGeometryGetName <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is string. It contains the name of a specified geometry.

Example

```
set geom_name [TdrGeometryGetName file1.tdr 1]
```

This example sets `geom_name` to the name of the specified geometry.

TdrGeometryGetNumRegion

Description

This command returns the number of regions in a geometry.

Syntax

```
TdrGeometryGetNumRegion <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is integer. It contains the number of regions in the specified geometry.

Example

```
set geom_num_region [TdrGeometryGetNumRegion file1.tdr 1]
```

This example sets `geom_num_region` to the number of regions in the specified geometry.

TdrGeometryGetNumState

Description

The command returns the number of geometry states.

Syntax

```
TdrGeometryGetNumState <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is integer. It contains the number of states of a specified geometry.

Example

```
set geom_num_state [TdrGeometryGetNumState file1.tdr 1]
```

This example sets `geom_num_state` to the number of states of the specified geometry.

TdrGeometryGetShift

Description

This command returns the shift of a geometry. The shift of a geometry is represented as a list of length 3.

Syntax

```
TdrGeometryGetShift <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is a list, which has the format {x y z}, which corresponds to the shifting values of the geometry.

Example

```
set geom_shift [TdrGeometryGetShift file1.tdr 1]
```

This example sets `geom_shift` to the shift list of the specified geometry.

TdrGeometryGetTagGroup

Description

This command returns the tag-group handle of a geometry. This handle can be used in the commands of the tag group-related functions and tag-related functions (see [Tag Group-Related Functions on page 83](#) and [Tag-Related Functions on page 93](#)).

Syntax

```
TdrGeometryGetTagGroup <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group-related functions and tag-related functions.

Example

```
set tg [TdrGeometryGetTagGroup file1.tdr 1]
```

This example sets `tg` to the tag-group handle of the specified geometry.

TdrGeometryGetTransform

Description

This command returns the transformation matrix of a geometry. The matrix size is 3×3 , and it is represented as a list.

Syntax

```
TdrGeometryGetTransform <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.

Return Value

Type of return value is a list, which has the format $\{x_{00}, x_{01}, x_{02}, \dots, x_{21}, x_{22}\}$, where x_{ij} is the element of the i -th row and j -th column of the transformation matrix.

Example

```
set geom_transformation [TdrGeometryGetTransform file1.tdr 1]
```

This example sets `geom_transformation` to the transformation list of the specified geometry.

TdrGeometryGetType

Description

This command returns the type of a geometry.

Syntax

```
TdrGeometryGetType <filename> <geometry_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.

Return Value

Type of return value is a string. It contains the type of a specified geometry. Possible values are:

- "envelop"
- "mixed_element"
- "tensor_uniform"
- "tensor_rectilinear"
- "tensor_warped"
- "tensor_xy"
- "grid_raytree"

Example

```
set geom_type [TdrGeometryGetType file1.tdr 1]
```

This example sets geom_type to the type of the specified geometry.

TdrGeometrySetName

Description

This command sets a new name for a geometry.

Syntax

```
TdrGeometrySetName <filename> <geometry_index> <name>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.
name	New name of geometry.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrGeometrySetName file1.tdr 1 new_geometry_name
```

This example assigns a new name `new_geometry_name` to the specified geometry.

TdrGeometrySetShift

Description

This command sets a new shift vector for a geometry. The shift of a geometry is represented as a list of length 3.

Syntax

```
TdrGeometrySetShift <filename> <geometry_index> <shift_list>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.
shift_list	List of length 3 in Tcl format that contains new shift vector.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
set new_shift {0.1 0.4 0.3}
TdrGeometrySetShift file1.tdr 1 $new_shift
```

This example assigns a new shift vector to the specified geometry.

TdrGeometrySetTransform

Description

This command sets a new rotation matrix for a geometry. The rotation matrix of a geometry is represented as a list of length 9.

Syntax

```
TdrGeometrySetTransform <filename> <geometry_index> <transformation_list>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} <$ number of geometries in the file.
transformation_list	List of length 9 in Tcl format that contains new rotation matrix. The order of elements is: $\{x_{00}, x_{01}, x_{02}, \dots, x_{21}, x_{22}\}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
set new_transform {1 3.14 4.13 3.13 1 6.13 4.13 6.13 1}  
TdrGeometrySetTransform file1.tdr 1 $new_transform
```

This example assigns the new rotation matrix:

```
1      3.14   4.13  
3.13   1      6.13  
4.13   6.13   1
```

to the specified geometry.

State-Related Functions

Table 6 lists all of the state-related Tcl commands that are available in Sentaurus Data Explorer.

Table 6 State-related functions of Tcl interface for TDR

Command	Description
TdrStateDelete	Deletes a specified state.
TdrStateGetName	Returns name of a state.
TdrStateGetTagGroup	Returns tag-group handle of the state.
TdrStateSetName	Sets new name for a specified state.

TdrStateDelete

Description

This command deletes a state.

NOTE When calling `TdrStateDelete` in a loop, you must traverse states in reverse order (see [Deleting Items in a Loop on page 21](#)).

Syntax

```
TdrStateDelete <filename> <geometry_index> <state_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is Boolean. It is `TRUE` if the operation is successful; otherwise, `FALSE`.

Example

```
TdrStateDelete file1.tdr 1 2
```

This example deletes the third state of the specified geometry.

TdrStateGetName

Description

This command returns the name of a state.

Syntax

```
TdrStateGetName <filename> <geometry_index> <state_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the name of a state for a geometry in a TDR file.

Example

```
set state_name [TdrStateGetName file1.tdr 1 2]
```

This example sets `state_name` to the name of the specified state.

TdrStateGetTagGroup

Description

This command returns the tag-group handle of a state. This handle can be used in the commands of the tag group-related functions and tag-related functions (see [Tag Group-Related Functions on page 83](#) and [Tag-Related Functions on page 93](#)).

Syntax

```
TdrStateGetTagGroup <filename> <geometry_index> <state_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
state_index	Index of a state for specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group-related functions and tag-related functions.

Example

```
set tg [TdrStateGetTagGroup file1.tdr 1 2]
```

This example sets `tg` to the tag-group handle of the specified state.

TdrStateSetName

Description

This command sets the name of a state.

Syntax

```
TdrStateSetName <filename> <geometry_index> <state_index> <name>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
name	New name of a state.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrStateSetName file1.tdr 1 2 FinalState
```

This example assigns the new name `FinalState` to the specified state.

Region-Related Functions

[Table 7](#) lists all of the region-related Tcl commands that are available in Sentaurus Data Explorer.

Table 7 Region-related functions of Tcl interface for TDR

Command	Description
TdrRegionGetDimension	Returns dimension of a region.
TdrRegionGetMaterial	Returns material of a region.
TdrRegionGetName	Returns a name of a region.
TdrRegionGetNumDataset	Return number of datasets for a region.
TdrRegionGetTagGroup	Returns tag-group handle of the region.
TdrRegionGetType	Returns type of a region.
TdrRegionSetMaterial	Sets new material for a region.
TdrRegionSetName	Sets new name for a region.

TdrRegionGetDimension

Description

This command returns the dimension of a region.

Syntax

```
TdrRegionGetDimension <filename> <geometry_index> <region_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is an integer. It contains the dimension of the specified region.

Example

```
set region_dim [TdrRegionGetDimension file1.tdr 1 2]
```

This example assigns to `region_dim` the dimension of the specified region.

TdrRegionGetMaterial

Description

This command returns the material of a region.

Syntax

```
TdrRegionGetMaterial <filename> <geometry_index> <region_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is a string. It contains the material of a region for a geometry in TDR file.

Example

```
set region_material [TdrRegionGetMaterial file1.tdr 1 2]
```

This example sets `region_material` to the material name of the specified region.

TdrRegionGetName

Description

This command returns the name of a region.

Syntax

```
TdrRegionGetName <filename> <geometry_index> <region_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is a string. It contains the name of a region for a geometry in a TDR file.

Example

```
set region_name [TdrRegionGetName file1.tdr 1 2]
```

This example sets `region_name` to the name of the specified region.

TdrRegionGetNumDataset

Description

This command returns the number of datasets for a region.

Syntax

```
TdrRegionGetNumDataset <filename> <geometry_index> <region_index>  
    <state_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is an integer. It contains the number of datasets of the specified region.

Example

```
set region_num_dataset [TdrRegionGetNumDataset file1.tdr 1 2 0]
```

This example sets `region_num_dataset` to the number of datasets of the specified region and state.

TdrRegionGetTagGroup

Description

This command returns the tag-group handle of a region. This handle can be used in the commands of the tag group-related functions and tag-related functions (see [Tag Group-Related Functions on page 83](#) and [Tag-Related Functions on page 93](#)).

Syntax

```
TdrRegionGetTagGroup <filename> <geometry_index> <region_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group-related functions and tag-related functions.

Example

```
set tg [TdrRegionGetTagGroup file1.tdr 1 2]
```

This example sets `tg` to the tag-group handle of the specified region.

TdrRegionGetType

Description

This command returns the type of a region.

Syntax

```
TdrRegionGetType <filename> <geometry_index> <region_index>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is a string. It contains the type of a region for a geometry in a TDR file. Possible values are:

- "bulk"
- "contact"
- "interface"
- "ten_bulk"
- "ten_contact"
- "ten_xy"
- "raytree"

Example

```
set region_type [TdrRegionGetType file1.tdr 1 2]
```

This example sets `region_type` to the type of the specified region.

TdrRegionSetMaterial

Description

This command sets the name of a material.

Syntax

```
TdrRegionSetMaterial <filename> <geometry_index> <region_index> <material>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
material	New material name for a region.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrRegionSetMaterial file1.tdr 1 2 Copper
```

This example assigns the new material name Copper to the specified region.

TdrRegionSetName

Description

This command sets the name for a region.

Syntax

```
TdrRegionSetName <filename> <geometry_index> <region_index> <name>
```

Arguments

Argument	Description
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
name	New name of a region.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrRegionSetName file1.tdr 1 2 new_region_name
```

This example assigns the new name `new_region_name` to the specified region.

Dataset-Related Functions

Table 8 lists all of the dataset-related Tcl commands that are available in Sentaurus Data Explorer.

Table 8 Dataset-related functions of Tcl interface for TDR

Command	Description
TdrDatasetDelete	Deletes dataset.
TdrDatasetDeleteByName	Deletes datasets by name.
TdrDatasetGetLocation	Returns location of a dataset.
TdrDatasetGetName	Returns name of a dataset.
TdrDatasetGetNumValue	Returns number of data values of a dataset.
TdrDatasetGetQuantity	Returns quantity of a dataset.
TdrDatasetGetStructure	Returns structure of a dataset.
TdrDatasetGetTagGroup	Returns tag-group handle of a dataset.
TdrDatasetGetType	Returns type of a dataset.
TdrDatasetGetUnit	Returns unit name of a dataset.
TdrDatasetGetUnitLong	Returns long unit name of a dataset.
TdrDatasetRename	Globally renames datasets.
TdrDatasetRenameQuantity	Globally changes the quantity of datasets.
TdrDatasetSetName	Sets new name for a region.
TdrDatasetSetQuantity	Sets new quantity for a region.

TdrDatasetDelete

Description

This command deletes a dataset.

NOTE When calling `TdrDatasetDelete` in a loop, you must traverse datasets in reverse order (see [Deleting Items in a Loop on page 21](#)).

Syntax

```
TdrDatasetDelete <filename> <geometry_index> <region_index> <state_index>  
                <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetDelete file1.tdr 1 2 0 0
```

This example deletes the first dataset of the specified region.

TdrDatasetDeleteByName

Description

This command deletes datasets by name.

Syntax

```
TdrDatasetDeleteByName <filename> [<name>]
```

Arguments

Argument	Description
filename	Name of a TDR file.
name	Name of datasets to be deleted. The name can take the form of a Tcl regular expression to specify which datasets should be deleted. If the argument is omitted, all datasets are deleted.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetDeleteByName file1.tdr DopingConcentration
```

This example deletes all datasets named `DopingConcentration` from the specified file.

TdrDatasetGetLocation

Description

This command returns the location of a dataset.

Syntax

```
TdrDatasetGetLocation <filename> <geometry_index> <region_index> <state_index>  
    <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the location of a dataset for a region in a TDR file. Possible values are: `vertex`, `edge`, `face`, `element`, `region`, `element_vertex`, `element_edge`, and `element_face`.

Example

```
set dataset_location [TdrDatasetGetLocation file1.tdr 1 2 0 0]
```

This example sets `dataset_location` to the location of the specified dataset.

TdrDatasetGetName

Description

This command returns the name of a dataset.

Syntax

```
TdrDatasetGetName <filename> <geometry_index> <region_index> <state_index>
<dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the name of a dataset for a region in a TDR file.

Example

```
set dataset_name [TdrDatasetGetName file1.tdr 1 2 0 0]
```

This example sets dataset_name to the name of the specified dataset.

TdrDatasetGetNumValue

Description

This command returns the number of data values of a dataset.

Syntax

```
TdrDatasetGetNumValue <filename> <geometry_index> <region_index> <state_index>  
    <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is an integer. It contains the number of data values of the specified dataset.

Example

```
set dataset_num_value [TdrDatasetGetNumValue file1.tdr 1 2 0 0]
```

This example sets `dataset_num_value` to the number of data values of the specified dataset.

TdrDatasetGetQuantity

Description

This command returns the quantity of a dataset.

Syntax

```
TdrDatasetGetQuantity <filename> <geometry_index> <region_index> <state_index>
<dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the quantity of a dataset for a region in a TDR file.

Example

```
set dataset_quantity [TdrDatasetGetQuantity file1.tdr 1 2 0 0]
```

This example sets dataset_quantity to the quantity of the specified dataset.

TdrDatasetGetStructure

Description

This command returns the structure of a dataset.

Syntax

```
TdrDatasetGetStructure <filename> <geometry_index> <region_index>  
    <state_index> <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the structure of a dataset for a region in a TDR file. Possible values are: scalar, vector, matrix, var_dim_vector, var_dim_matrix, and tensor_sym.

Example

```
set dataset_structure [TdrDatasetGetStructure file1.tdr 1 2 0 0]
```

This example sets dataset_structure to the structure of the specified dataset.

TdrDatasetGetTagGroup

Description

This command is used to obtain the tag-group handle of a dataset. This handle can be used in the commands of the tag group-related functions and tag-related functions (see [Tag Group-Related Functions on page 83](#) and [Tag-Related Functions on page 93](#)).

Syntax

```
TdrDatasetGetTagGroup <filename> <geometry_index> <region_index> <state_index>
<dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group-related functions and tag-related functions.

Example

```
set tg [TdrDatasetGetTagGroup file1.tdr 1 2 0 0]
```

This example sets `tg` to the tag-group handle of the specified dataset.

TdrDatasetGetType

Description

This command returns the type of a dataset.

Syntax

```
TdrDatasetGetType <filename> <geometry_index> <region_index> <state_index>  
    <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. It contains the type of a dataset for a region in a TDR file. Possible values are:

- "vti" (integer)
- "vtf" (float)
- "vtd" (double)
- "vtcf" (complex float)
- "vtcd" (complex double)

Example

```
set dataset_type [TdrDatasetGetType file1.tdr 1 2 0 0]
```

This example sets dataset_type to the type of the specified dataset.

TdrDatasetGetUnit, TdrDatasetGetUnitLong

Description

The `TdrDatasetGetUnit` command is used to obtain the unit name of a dataset. For the long name of a unit, use the `TdrDatasetGetUnitLong` command.

Syntax

```
TdrDatasetGetUnit <filename> <geometry_index> <region_index> <state_index>
<dataset_index>
```

```
TdrDatasetGetUnitLong <filename> <geometry_index> <region_index> <state_index>
<dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a string. For the `TdrDatasetGetUnit` command, it contains the unit name of a dataset. For the `TdrDatasetGetUnitLong` command, it contains the long unit name.

Example

```
set dataset_name [TdrDatasetGetUnit file1.tdr 1 2 0 0]
set dataset_long_name [TdrDatasetGetUnitLong file1.tdr 1 2 0 0]
```

This example sets `dataset_name` to the name of the unit and `dataset_long_name` to the long name of the unit of the specified dataset.

TdrDatasetRename

Description

This command performs a global (file level) search-and-replace operation for dataset names.

Syntax

```
TdrDatasetRename <filename> [<old-name>] <new-name>
```

Arguments

Argument	Description
filename	Name of a TDR file.
new-name	New dataset name.
old-name	Old dataset name. The name can take the form of a Tcl regular expression to specify which datasets should be renamed. If the argument is omitted, all datasets are renamed.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetRename file1.tdr draincurrent {Drain Current}
```

This example assigns the new name `Drain Current` to all datasets named `draincurrent`.

TdrDatasetRenameQuantity

Description

This command performs a global (file level) search-and-replace operation for dataset quantities.

Syntax

```
TdrDatasetRenameQuantity <filename> [<old-quantity>] <new-quantity>
```

Arguments

Argument	Description
filename	Name of a TDR file.
new-quantity	New dataset quantity.
old-quantity	Old dataset quantity. The quantity can take the form of a Tcl regular expression to specify which dataset quantities should be renamed. If the argument is omitted, all dataset quantities are renamed.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetRenameQuantity file1.tdr Arsenic ArsenicConcentration
```

This example assigns the new quantity `ArsenicConcentration` to all datasets with the quantity `Arsenic`.

TdrDatasetSetName

Description

This command sets the name of a dataset.

Syntax

```
TdrDatasetSetName <filename> <geometry_index> <region_index> <state_index>  
    <dataset_index> <name>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
name	New name of a dataset.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetSetName file1.tdr 1 2 0 0 new_dataset_name
```

This example assigns the new name `new_dataset_name` to the specified dataset.

TdrDatasetSetQuantity

Description

This command sets the quantity for a dataset.

Syntax

```
TdrDatasetSetQuantity <filename> <geometry_index> <region_index> <state_index>
    <dataset_index> <quantity>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
quantity	New quantity of a dataset. Valid quantities are defined by the DATEX standard.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDatasetSetQuantity file1.tdr 1 2 0 0 ElectrostaticPotential
```

This example assigns a new quantity `ElectrostaticPotential` to the specified dataset.

3: Tcl Interface

Data Value–Related Functions

Data Value–Related Functions

Table 9 lists all of the data value–related Tcl commands that are available in Sentaurus Data Explorer.

Table 9 Data value–related functions of Tcl interface for TDR

Command	Description
TdrDataGetAllCoordinates	Returns a list of all coordinates for all data vertices corresponding to a dataset.
TdrDataGetComponent	Returns component of a specified data value for given row and column.
TdrDataGetCoordinate	Returns coordinates of a data value.
TdrDataGetNumCol	Returns number of columns for the specified data value.
TdrDataGetNumRow	Returns number of rows for the specified data value.
TdrDataGetValue	Returns data value for scalar type of a dataset.
TdrDataSetComponent	Sets component of a specified data value for given row and column.

TdrDataGetAllCoordinates

Description

This command returns a list of all coordinates for all data vertices corresponding to a dataset.

Syntax

```
TdrDataGetAllCoordinates <filename> <geometry_index> <region_index>
    <state_index> <dataset_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a list, which contains all coordinates of all the data vertices. The list has the format:

$$\{ c_{0,0}, \dots, c_{0,N} \} \{ c_{1,0}, \dots, c_{1,N} \} \dots \{ c_{M,0}, \dots, c_{M,N} \}$$

where N is the geometry dimension (1, 2, or 3), and M is the number of data points in the dataset.

Example

```
set coords [TdrDataGetAllCoordinates file1.tdr 1 2 0 0 15]
```

This example sets `coords` to a list of all coordinates of the specified dataset. For example, if the geometry dimension is 3 and there are two data points with coordinates (0, 0, 0) and (1, 1, 1), respectively, the result will be (a list consisting of two lists with a length of 3):

$$\{ 0 \ 0 \ 0 \} \{ 1 \ 1 \ 1 \}$$

TdrDataGetComponent

Description

This command returns a data component with a specified index from a dataset. It supports all data structure types: scalar, vector, matrix, var_dim_vector, var_dim_matrix, and tensor_sym.

Syntax

```
TdrDataGetComponent <filename> <geometry_index> <region_index> <state_index>
<dataset_index> <value_index> [<row>] [<col>]
```

Arguments

Argument	Description
col	Index of a column in a data value. Requires $0 \leq \text{col} < \text{number of columns in the data value}$. If the column is not specified, it is set to 0. It is not necessary and is ignored for data values of structure: scalar, vector, and vector with variable dimension.
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
row	Index of a row of a data value. Requires $0 \leq \text{row} < \text{number of rows in the data value}$. If the row is not specified, it is set to 0. It is not necessary and is ignored for scalar data values.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
value_index	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

Type of return value is a string containing the specified data component. Depending on the value type of the dataset, the returned component is either an integer, a float, or a double.

Example

```
set data_component [TdrDataGetComponent file1.tdr 1 2 0 0 15 1 2]
```

This example sets `data_component` to the value of the specified component.

TdrDataGetCoordinate

Description

This command returns a coordinate corresponding to a data value.

Syntax

```
TdrDataGetCoordinate <filename> <geometry_index> <region_index> <state_index>  
    <dataset_index> <value_index> <coordinate_index>
```

Arguments

Argument	Description
coordinate_index	Index of a coordinate. Requires $0 \leq \text{coordinate_index} < \text{dimension of the geometry}$.
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
value_index	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

The specified coordinate is returned as a double.

Example

```
set x [TdrDataGetCoordinate file1.tdr 1 2 0 0 15 1 0]
```

This example sets x to the first coordinate of the specified dataset value.

TdrDataGetNumCol

Description

This command returns the number of columns of a dataset value.

Syntax

```
TdrDataGetNumCol <filename> <geometry_index> <region_index> <state_index>
<dataset_index> <value_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
value_index	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

Type of return value is an integer. It contains the number of columns of a dataset value. For the structure types scalar, vector, and vector with variable dimension, it is always 1.

Example

```
set data_value_num_col [TdrDataGetNumCol file1.tdr 1 2 0 0 15]
```

This example sets data_value_num_col to the number of columns of the specified data value.

TdrDataGetNumRow

Description

This command returns the number of rows of a dataset value.

Syntax

```
TdrDataGetNumRow <filename> <geometry_index> <region_index> <state_index>  
  <dataset_index> <value_index>
```

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
value_index	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

Type of return value is an integer. It contains the number of rows of a dataset value. For values of structure type scalar, it is always 1.

Example

```
set data_value_num_row [TdrDataGetNumRow file1.tdr 1 2 0 0 15]
```

This example sets data_value_num_row to the number of rows of the specified data value.

TdrDataGetValue

Description

This command returns a single data value from a scalar dataset. For non-scalar datasets, use the `TdrDataGetComponent` command (see [TdrDataGetComponent](#) on page 76).

Syntax

```
TdrDataGetValue <filename> <geometry_index> <region_index> <state_index>
<dataset_index> <value_index>
```

Arguments

Argument	Description
<code>dataset_index</code>	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
<code>filename</code>	Name of a TDR file.
<code>geometry_index</code>	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
<code>region_index</code>	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
<code>state_index</code>	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
<code>value_index</code>	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

Type of return value is a string containing the specified data value. Depending on the value type of the dataset, the returned component is either an integer, a float, or a double.

Example

```
set data_value [TdrDataGetValue file1.tdr 1 2 0 0 15]
```

This example sets `data_value` to the specified dataset value.

TdrDataSetComponent

Description

This command sets a data component with a specified index from a dataset. It supports all data structure types: `scalar`, `vector`, `matrix`, `var_dim_vector`, `var_dim_matrix`, and `tensor_sym`.

Syntax

```
TdrDataSetComponent <filename> <geometry_index> <region_index> <state_index>
  <dataset_index> <value_index> [<row>] [<col>] <value>
```

Arguments

Argument	Description
<code>col</code>	Index of a column in a data component. Requires $0 \leq \text{col} < \text{number of columns in the data component}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for the data structures of type: <code>scalar</code> , <code>vector</code> , and <code>vector with variable dimension</code> .
<code>dataset_index</code>	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
<code>filename</code>	Name of a TDR file.
<code>geometry_index</code>	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
<code>region_index</code>	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
<code>row</code>	Index of a row of a data component. Requires $0 \leq \text{row} < \text{number of rows in the data component}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for scalar data structures.
<code>state_index</code>	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.
<code>value</code>	A value to be set up for a specified component. Possible types are integer, float, and double.
<code>value_index</code>	Index of a data value for the specified dataset. Requires $0 \leq \text{value_index} < \text{number of data values in the dataset}$.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrDataSetComponent file1.tdr 1 2 0 0 15 1 2 3.1415926
```

This example assigns the value "3.1415926" to the specified data component.

Tag Group–Related Functions

[Table 10](#) lists all of the tag group–related Tcl commands that are available in Sentaurus Data Explorer.

Table 10 Tag group–related functions of Tcl interface for TDR

Command	Description
TdrTagGroupCreate	Creates a new tag group.
TdrTagGroupDelete	Deletes tag group using its index.
TdrTagGroupDeleteByName	Deletes tag group using its name.
TdrTagGroupGetByPath	Returns handle of the tag group given by path in the TDR hierarchy.
TdrTagGroupGetName	Returns name of a tag group.
TdrTagGroupGetNumTag	Returns number of tags that contain the tag group.
TdrTagGroupGetNumTagGroup	Returns number of tag groups that contain the specified tag group.
TdrTagGroupGetTagGroup	Returns TagGroup handle of the tag group.

TdrTagGroupCreate

Description

This command creates a tag group.

Syntax

```
TdrTagGroupCreate <parent_tag_group> <name>
```

Arguments

Argument	Description
name	Name of the tag group to be created.
parent_tag_group	Handle of tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group–related functions and tag-related functions.

Example

```
TdrTagGroupCreate [TdrFileGetTagGroup file1.tdr] WaferDetails
```

This example creates a new tag group named `WaferDetails` inside the root tag group of the specified file.

TdrTagGroupDelete

Description

This command deletes a tag group.

NOTE When calling `TdrTagGroupDelete` in a loop, you must traverse tag groups in reverse order (see [Deleting Items in a Loop on page 21](#)).

Syntax

```
TdrTagGroupDelete <parent_tag_group> <tag_group_index>
```

Arguments

Argument	Description
parent_tag_group	Handle of tag group. It can be obtained only from <code>Tdr<T>GetTagGroup</code> commands, where <i>T</i> can be: <code>File</code> , <code>Geometry</code> , <code>State</code> , <code>Region</code> , <code>Dataset</code> , and <code>TagGroup</code> , as well as from the <code>TdrTagGroupCreate</code> and <code>TdrTagGroupGetByPath</code> commands.
tag_group_index	Index of tag group which is to be deleted.

Return Value

Type of return value is Boolean. It is `TRUE` if the operation is successful; otherwise, `FALSE`.

Example

```
TdrTagGroupDelete [TdrFileGetTagGroup file1.tdr] 0
```

This example deletes the first tag group at the file level.

TdrTagGroupDeleteByName

Description

This command deletes a tag group using its name.

Syntax

```
TdrTagGroupDeleteByName <parent_tag_group> <tag_group_name>
```

Arguments

Argument	Description
parent_tag_group	Handle of parent tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_group_name	Name of tag group to be deleted.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
TdrTagGroupDeleteByName [TdrTagGroupGetByPath file1.tdr 0 0 state/KMC] Defects
```

This example deletes the tag group named `Defects` from the KMC tag group of the specified state.

TdrTagGroupGetByPath

Description

This command returns the handle of a tag group specified by its absolute path in the TDR hierarchy.

Since tag-group hierarchies can be attached to different types of TDR object, the first element of the path (the root specifier) is used to specify the type of TDR object that is at the root of the path. The actual instance of the TDR object is specified by a set of indices, according to the type of the root TDR object. The root specifier can be an empty string or one of the strings *geometry*, *region*, *state*, *dataset*. The empty root specifier is used for the tag-group hierarchy, which is attached at the file level, also called *collection level*.

Syntax

General:

```
TdrTagGroupGetByPath <filename> [<geometry_index>] [<region_index>]
    [<state_index>] [<dataset_index>] <path>
```

```
path: [<root-object-specifier>]/<tag-group-name>/<tag-group-name>/...
```

For tag-group hierarchies attached to the file:

```
TdrTagGroupGetByPath <filename> <tag-group>/<tag-group>/...
```

For tag-group hierarchies attached to a geometry:

```
TdrTagGroupGetByPath <filename> <geometry_index>
    geometry/<tag-group>/<tag-group>/...
```

For tag-group hierarchies attached to region:

```
TdrTagGroupGetByPath <filename> <geometry_index> <region_index>
    region/<tag-group>/<tag-group>/...
```

For tag-group hierarchies attached to a state:

```
TdrTagGroupGetByPath <filename> <geometry_index> <state_index>
    state/<tag-group>/<tag-group>/...
```

For tag-group hierarchies attached to a dataset:

```
TdrTagGroupGetByPath <filename> <geometry_index> <region_index> <state_index>
    <dataset_index> dataset/<tag-group>/<tag-group>/...
```

3: Tcl Interface

Tag Group–Related Functions

Arguments

Argument	Description
dataset_index	Index of a dataset for a specified geometry and region in TDR file. Requires $0 \leq \text{dataset_index} < \text{number of datasets for the specified region}$.
filename	Name of a TDR file.
geometry_index	Index of geometry in TDR file. Requires $0 \leq \text{geometry_index} < \text{number of geometries in the file}$.
path	Path of the tag group, consisting of an initial root specifier and tag-group names separated by the "/" character.
region_index	Index of a region for a specified geometry in TDR file. Requires $0 \leq \text{region_index} < \text{number of regions for the specified geometry}$.
state_index	Index of a state for a specified geometry in TDR file. Requires $0 \leq \text{state_index} < \text{number of states for the specified geometry}$.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group–related functions and tag-related functions.

Example

```
set tg [TdrTagGroupGetByPath file1.tdr 0 0 state/KMC/Defects]
```

This example sets `tg` to the handle of the tag group `Defects`, which is attached to the first state of the first geometry.

TdrTagGroupName

Description

This command returns the name of a tag group.

Syntax

```
TdrTagGroupName <tag_group>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.

Return Value

Type of return value is a string. It contains the name of the tag group. If the tag_group argument refers to the root tag group of a TDR object, the returned name is either collection, geometry, region, state, or dataset, depending on the type of TDR object.

Example

```
set taggroup_name [TdrTagGroupName [TdrFileGetTagGroup file1.tdr]]
```

This example sets taggroup_name to the name of the root tag group of the specified file.

TdrTagGroupGetNumTag

Description

This command returns the number of tags in the specified tag group.

Syntax

```
TdrTagGroupGetNumTag <tag_group>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.

Return Value

Type of return value is an integer. It contains the number of tags in the specified tag group.

Example

```
set tag_num [TdrTagGroupGetNumTag [TdrFileGetTagGroup file1.tdr]]
```

This example sets `tag_num` to the number of tags in the root tag group of the specified file.

TdrTagGroupGetNumTagGroup

Description

This command returns the number of tag groups in a tag group.

Syntax

```
TdrTagGroupGetNumTagGroup <tag_group>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.

Return Value

Type of return value is an integer. It contains the number of tag groups in the specified tag group.

Example

```
set tag_group_num [TdrTagGroupGetNumTagGroup [TdrFileGetTagGroup file1.tdr]]
```

This example sets `tag_group_num` to the number of tag groups in the root tag group of the specified file.

TdrTagGroupGetTagGroup

Description

This command returns the handle of a tag group contained in the specified tag group.

Syntax

```
TdrTagGroupGetTagGroup <parent_tag_group> <tag_group_index>
```

Arguments

Argument	Description
parent_tag_group	Handle of tag group. It can be obtained only from Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_group_index	Index of a tag group in the specified tag group. Requires $0 \leq \text{tag_group_index} < \text{number of tag groups in the specified tag group}$.

Return Value

Type of return value is a handle. It can be used only in the commands of the tag group–related functions and tag-related functions.

Example

```
set tg [TdrTagGroupGetByPath file1.tdr 0 0 state/KMC]
set tg0 [TdrTagGroupGetTagGroup $tg 0]
```

This example sets `tg0` to the handle of the first tag group inside the `state/KMC` tag group.

Tag-Related Functions

Table 11 lists all of the tag-related Tcl commands that are available in Sentaurus Data Explorer.

Table 11 Tag-related functions of Tcl interface for TDR

Command	Description
TdrTagCreateScalar	Creates a new tag with a scalar structure.
TdrTagDelete	Deletes a tag.
TdrTagDeleteByName	Deletes a tag with the given name.
TdrTagGetComponent	Returns component with a specified column and row of a tag.
TdrTagGetName	Returns name of a tag.
TdrTagGetNumCol	Returns number of columns in a tag.
TdrTagGetNumRow	Returns number of rows in a tag.
TdrTagGetStructure	Returns structure of a tag.
TdrTagGetType	Returns type of a tag.
TdrTagGetValue	Returns value of a tag.
TdrTagSetComponent	Sets component of a tag.

TdrTagCreateScalar

Description

This command creates a new scalar tag.

Syntax

```
TdrTagCreateScalar <parent_tag_group> <name> <type> <value>
```

Arguments

Argument	Description
name	Name of new tag.
parent_tag_group	Handle of parent tag group. For this tag group, it creates a scalar tag. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
type	Type of new tag. Possible values: "vtb" (Boolean), "vti32" (integer 32 bits), "vti64" (integer 64 bits), "vtd" (double), "vts" (string), and "vtf" (float).
value	Value of new tag. It should match the datatype.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
TdrTagCreateScalar $tg MyPi vtd 3.1415926
```

This example creates a tag named `MyPi` inside the root tag group of the specified file. Its type is `vtd` (double), and its value is `3.1415926`.

TdrTagDelete

Description

This command deletes a tag.

NOTE When calling `TdrTagDelete` in a loop, you must traverse tags in reverse order (see [Deleting Items in a Loop on page 21](#)).

Syntax

```
TdrTagDelete <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the <code>Tdr<T>GetTagGroup</code> commands, where <i>T</i> can be: <code>File</code> , <code>Geometry</code> , <code>State</code> , <code>Region</code> , <code>Dataset</code> , and <code>TagGroup</code> , as well as from the <code>TdrTagGroupCreate</code> and <code>TdrTagGroupGetByPath</code> commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} <$ number of tags in the specified tag group.

Return Value

Type of return value is Boolean. It is `TRUE` if the operation is successful; otherwise, `FALSE`.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
TdrTagDelete $tg 2
```

This example deletes the third tag from the root tag group of the specified file.

TdrTagDeleteByName

Description

This command deletes a tag with a given name.

Syntax

```
TdrTagDeleteByName <tag_group> <tag_name>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_name	Name of a tag in the specified tag group.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
TdrTagDeleteByName $tg Temp
```

This example deletes the tag named Temp from the root tag group of the specified file.

TdrTagGetComponent

Description

This command returns the component of a tag.

Syntax

```
TdrTagGetComponent <tag_group> <tag_index> [<row>] [<col>]
```

Arguments

Argument	Description
col	Index of a column of a tag component. Requires $0 \leq \text{col} < \text{number of columns of a tag}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for scalar and vector tag structures.
row	Index of a row of a tag component. Requires $0 \leq \text{row} < \text{number of rows of a tag}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for scalar data structures.
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} < \text{number of tags in the specified tag group}$.

Return Value

Type of return value is a string. It contains the component of a tag with specified row and column indices. Possible value types are: Boolean, integer 32 bits, integer 64 bits, float, double, and string.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_component [TdrTagGetComponent $tg 2 1 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain a component of the second row and the third column of the third tag in this tag group, which is stored in `tag_component`.

TdrTagGetName

Description

This command returns the name of a tag.

Syntax

```
TdrTagGetName <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} <$ number of tags in the specified tag group.

Return Value

Type of return value is a string. It contains the name of a tag.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_name [TdrTagGetName $tg 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain the name of the third tag in this tag group, which is stored in `tag_name`.

TdrTagGetNumCol

Description

This command returns the number of columns of a tag.

Syntax

```
TdrTagGetNumCol <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} < \text{number of tags in the specified tag group}$.

Return Value

Type of return value is an integer. It contains the number of columns in a tag. For scalar and vector tag structures, it is always 1.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_num_col [TdrTagGetNumCol $tg 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain the number of columns of the third tag in this tag group, which is stored in `tag_num_col`.

TdrTagGetNumRow

Description

This command returns the number of rows of a tag.

Syntax

```
TdrTagGetNumRow <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} <$ number of tags in the specified tag group.

Return Value

Type of return value is an integer. It contains the number of rows in a tag. For scalar tag structures, it is always 1.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_num_row [TdrTagGetNumRow $tg 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain the number of rows of the third tag in this tag group, which is stored in `tag_num_row`.

TdrTagGetStructure

Description

This command returns the structure of a tag.

Syntax

```
TdrTagGetStructure <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where T can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} <$ number of tags in the specified tag group.

Return Value

Type of return value is a string. It contains the structure of a tag. Possible values are:

- "scalar"
- "vector"
- "matrix"

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_structure [TdrTagGetStructure $tg 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain the structure of the third tag in this tag group, which is stored in `tag_structure`.

TdrTagGetType

Description

This command returns the type of a tag.

Syntax

```
TdrTagGetType <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} <$ number of tags in the specified tag group.

Return Value

Type of return value is a string. It contains the structure of a tag. Possible values are:

- "v**t**b" (Boolean)
- "v**t**i32" (integer 32 bits)
- "v**t**i64" (integer 64 bits)
- "v**t**d" (double)
- "v**t**cd" (complex double)
- "v**t**s" (string)
- "v**t**f" (float)

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_type [TdrTagGetType $tg 2]
```

This example sets `tg` to the tag group handle of the file named `file1.tdr`. Then, this handle is used to obtain the type of the third tag in this tag group, which is stored in `tag_type`.

TdrTagGetValue

Description

This command returns the value of a tag. This command supports only tags with scalar structures. For other tag structures, use the `TdrTagGetComponent` command (see [TdrTagGetComponent on page 97](#)).

Syntax

```
TdrTagGetValue <tag_group> <tag_index>
```

Arguments

Argument	Description
tag_group	Handle of tag group. It can be obtained only from the <code>Tdr<T>GetTagGroup</code> commands, where <i>T</i> can be: <code>File</code> , <code>Geometry</code> , <code>State</code> , <code>Region</code> , <code>Dataset</code> , and <code>TagGroup</code> , as well as from the <code>TdrTagGroupCreate</code> and <code>TdrTagGroupGetByPath</code> commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} < \text{number of tags in the specified tag group}$.

Return Value

Type of return value is a string. It contains the value of a tag.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
set tag_value [TdrTagGetValue $tg 2]
```

This example sets `tg` to the tag-group handle of the file named `file1.tdr`. Then, this handle is used to obtain the value of the third tag in this tag group, which is stored in `tag_value`.

TdrTagSetComponent

Description

This command sets a component of a tag.

Syntax

```
TdrTagSetComponent <tag_group> <tag_index> [<row>] [<col>] <value>
```

Arguments

Argument	Description
col	Index of a column of a tag component. Requires $0 \leq \text{col} < \text{number of columns of a tag}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for scalar and vector tag structures.
row	Index of a row of a tag component. Requires $0 \leq \text{row} < \text{number of rows of a tag}$. If the value is not specified, it is set to 0. It is not necessary and is ignored for scalar data structures.
tag_group	Handle of tag group. It can be obtained only from the Tdr<T>GetTagGroup commands, where <i>T</i> can be: File, Geometry, State, Region, Dataset, and TagGroup, as well as from the TdrTagGroupCreate and TdrTagGroupGetByPath commands.
tag_index	Index of the tag in the specified tag group. Requires $0 \leq \text{tag_index} < \text{number of tags in the specified tag group}$.
value	New value of a tag component. Possible value types are: Boolean, integer 32 bits, integer 64 bits, float, double, and string.

Return Value

Type of return value is Boolean. It is TRUE if the operation is successful; otherwise, FALSE.

Example

```
set tg [TdrFileGetTagGroup file1.tdr]
TdrTagSetComponent $tg 0 1 2 3.1415926
```

This example sets component (1,2) of the specified tag to 3.1415926. Note that the row and column indices start from zero.

CHAPTER 4 Reference Guide

This chapter provides reference material for using Sentaurus Data Explorer.

Environment Variables and Configuration Files

The environment variable `TDFLIB` is used to locate a directory that contains the following configuration files:

- `mtx.db` is used to convert material names from TDF and TIF files to TDR material names.
- `sol.db` is used to convert quantity names and conversion factors from TDF and TIF to TDR.
- `mat.dbs` and `sol.dbs` are required only for conversion of TIF files.

The `datexcodes.txt` file is used to convert TDR files to TIF files.

See [Utilities User Guide, Chapter 1 on page 1](#), for more information on the `datexcodes.txt` file and the search strategy.

Supported Conversions

For all conversions, the internal representation uses the TDR format.

[Table 12](#) and [Table 13 on page 106](#) show which file format conversions are supported: ++ indicates that no information is lost during the conversion and + indicates that conversion is possible but some information contained in the source file may not be converted.

Table 12 Supported conversions of grid and data files

Input	Output		
	TDR	DF-ISE	TIF
TDR		+	+
DF-ISE	++		+
TDF	+	+	+
TIF	+	+	

Table 13 Supported conversions of xy files

Input	Output	
	TDR	DF-ISE
TDR		+
DF-ISE	++	
IVL	+	+
PLX	+	+

TDF-to-TDR Conversions

TDF Format Constraints

Sentaurus Data Explorer only converts TDF files that contain a finite-element grid. Other possible contents such as tensor grids or boundaries are not supported by the converter.

Material Names

The conversion of material names is based on the entries in the `DFISEName` column in the `mtr.db` file. Users can create a local copy of the `mtr.db` file, and add or modify the `DFISEName` entries if the provided values are not appropriate.

If the name of a material cannot be found in the `mtr.db` file, a warning is displayed and the material name is not modified by Sentaurus Data Explorer.

If the material is found in the `mtr.db` file and there is a `DFISEName` entry, this one is used. If there is no `DFISEName` entry, the root material is used and converted to the corresponding `DFISEName`, and a warning is displayed.

Quantity Names

The conversion of quantity names is based on the entries in the `DatexName` column in the `sol.db` file. Users can create a local copy of the `sol.db` file, and add or modify the `DatexName` entries if the provided values are not appropriate.

If the name of a quantity cannot be found in the `sol.db` file, a warning is displayed and the quantity name is not modified by Sentaurus Data Explorer.

If the quantity is found in the `sol.db` file and there is a `DateXName` entry, this one is used. If there is no `DateXName` entry, the `PrintName` is used and a warning is displayed.

Conversion Factor

For some of the TDF quantities, the units differ from the corresponding TDR quantities and an appropriate conversion factor must be applied. The value of the conversion factor for a quantity is taken from the `convFac` column in the `sol.db` file.

Ignoring Unknown Quantities

Sentaurus Device cannot handle datasets with quantities for which there is no corresponding DATEX name. With the option `-q`, it is possible to ignore these datasets during conversion.

Electrodes and Thermodes

TDF files can contain electrodes and thermodes for electrical and thermal contacts, respectively. In TDR and DF-ISE, there is only one type of contact and it is necessary to specify in the command file of Sentaurus Device the type of boundary condition for which a contact is used.

Volume Regions With Material Electrode or Thermode

Volume regions are regions that have the same dimension as the geometry to which they belong. In TDR, the dimension of contact regions is one less than the dimension of the geometry to which they belong. Therefore, TDF volume regions with material electrodes or thermodes cannot be converted into TDR contacts. These volume regions are ignored during conversion and a warning is displayed.

In Taurus™ Process, the material can be changed to avoid ignoring these regions during conversion:

```
RedefineRegion(name=... newMaterial=...)
```

or:

```
RedefineRegion(material=... newMaterial=...)
```

To create TDR contacts during conversion, it is necessary to wrap these volume regions in Taurus Process in a surface contact:

```
defineContact(region=... name=...)
```

Removing Ambient Regions

Structures saved by Taurus Process contain ambient regions, which are usually unwanted for device simulations. Using the option `-a`, it is possible to ignore ambient regions during conversion.

Interface Regions

Conversion of interface regions is not supported.

Inconsistent Faces

Taurus Process and Taurus Device cannot assemble the equation system for pyramid elements. Therefore, pyramids are split into tetrahedrons. However, the element adjacent to the rectangular face of the pyramid is not split. Inconsistent faces are the result because, after the split, two triangular faces are adjacent to a rectangular face.

In previous versions of Taurus Process, pyramids were always split. The default behavior of Taurus Process has been changed to keep pyramids. However, when simulating a diffusion step, the pyramids are split and inconsistent faces are generated. Different possibilities to make the faces consistent are described here.

Taurus Device will always split pyramids into tetrahedrons and save files containing inconsistent faces. These files can be loaded into Taurus Process and made consistent.

In Taurus Process, different solutions are possible, depending on the application:

- Prevent Taurus Process from splitting elements and creating inconsistent faces. The following command will ensure that only consistent mixed-element meshes are built (starting from the next regridting):

```
refinements(regrid(splitPyramids=false))
```

- Make faces consistent. Making a mesh consistent will create pyramids and it may add a few points. It is better to generate a consistent mesh in the first instance, keeping the pyramids. Making an inconsistent mesh consistent is intended to be an emergency procedure that needs to be used, for example, when an inconsistent mesh is loaded and regridting should

be avoided. The following command will change an inconsistent mesh into a consistent one during a simulation in Taurus Process. This is a one-time operation, until the next regridting:

```
redefineDevice(consistentFaces)
```

The following command will make an inconsistent mesh consistent at the time of saving a file, while staying inconsistent during the further Taurus Process simulation:

```
save(... consistentFaces)
```

Either of the above commands can be used immediately after, for example, a `defineDevice(meshfile=...)` command that loads a file.

- Force Taurus Process to split all 3D elements into tetrahedrons before saving a TDF file. Taurus Process will always build simplex meshes starting with the next regrid after the user specifies:

```
refinements(regrid(simplexMesh))
```

Taurus Process will convert the existing mesh in memory into a simplex mesh (this is a one-time operation; in the next regrid, it will not build simplex meshes):

```
redefineDevice(simplexMesh)
```

Taurus Process will convert the mesh into a simplex mesh when saving a file (but will remain with whatever meshes it has in memory and continue building such meshes in subsequent regrids):

```
save(... simplexMesh)
```

The last two commands can be used immediately after:

```
defineDevice(meshfile=...)
```

to convert a mesh from a loaded TDF file into a simplex mesh.

NOTE Conversion into a simplex mesh will considerably increase the number of elements, which can lead to a slowdown of subsequent Sentaurus Device simulations.

Splitting Rectangles

Sentaurus Process can only use grids that contain triangles or tetrahedrons. Two-dimensional grids saved by Taurus Process can contain triangles and rectangles. Using the option `-r`, the rectangles can be split into triangles during conversion.

For 3D grids, a similar splitting of elements into tetrahedrons is necessary. However, this is not supported by Sentaurus Data Explorer and must be performed by Taurus Process.

Extracting Boundaries

Sentaurus Data Explorer only converts a TDF grid to a TDR grid. To remesh the structure, it is necessary to extract and simplify the boundary. Extraction of the boundary can be performed by using Sentaurus Mesh. Simplification of the extracted boundary can be performed by using Sentaurus Structure Editor.

TIF-to-TDR Conversions

Material and Quantity Names

TIF material and quantity names are first converted to the corresponding TDF material and quantity names. Then, the same procedure as previously described for TDF material and quantity names is used. See [Material Names on page 106](#) and [Quantity Names on page 106](#).

Removing Contact Regions

TIF files frequently contain volume regions for which the material or root material is `conductor` and contact regions, which are the boundary of these volume regions. With the option `-c`, it is possible to ignore the volume regions and keep only the boundary regions that can then be used in Sentaurus Device.

Missing Ambient Regions

Structures saved by TSUPREM-4 do not contain ambient regions. These structures cannot be used by Sentaurus Process without adding a gas region. A possible solution is to deposit an otherwise unused material as the last process step before writing a TIF file that will be converted for use with Sentaurus Process and to rename the material to `Gas`.

TDR-to-TIF Conversions

Material and Quantity Names

The conversion of material and quantity names is based on the information contained in the `alter1` entries in the `datexcodes.txt` file. Users can create a local copy of the `datexcodes.txt` file, and add or modify the `alter1` entries if the provided values are not appropriate.

If there is no `alter1` entry, the material or quantity name is not changed and a warning is displayed.

Contacts

Contacts are converted into regions with the material `Elec`.

Interface Regions

Conversion of interface regions is not supported.

Region Names

Space characters in region names are replaced by underscores.

Mirroring

Number of Regions

Regions that touch the mirror axis are merged with their mirror image. Therefore, the number of regions in the new geometry is less than twice the number of regions in the original structure.

Naming Regions

By default, the name of new regions is the name of the original region with the suffix `_mirrored`. It is possible to rename new regions automatically by specifying the name of the original region and the name of the new region.

Vector Datasets

For vectors located on the mirror axis, the value of the component perpendicular to the mirror axis is set to zero. For vectors that are not located on the mirror axis, the sign of the vector component perpendicular to the mirror axis is inverted.

APPENDIX A Tcl Commands Interface of Sentaurus Data Explorer

This appendix provides a full list of the Tcl commands of Sentaurus Data Explorer as well as the returned values and all parameters.

File (TDR Collection) Commands

Boolean	TdrFileClose	<filename>
Boolean	TdrFileConvert	<convert-command> [parameter] <source-file> [<destination-file>]
Integer	TdrFileGetNumGeometry	<filename>
Handle	TdrFileGetTagGroup	<filename>
Boolean	TdrFileOpen	<filename> [-native_units] [-reference_coordinates]
Boolean	TdrFileSave	<filename> [<new_filename>]

Geometry Commands

Boolean	TdrGeometryDelete	<filename> <geometry_index>
Integer	TdrGeometryGetDimension	<filename> <geometry_index>
String	TdrGeometryGetName	<filename> <geometry_index>
Integer	TdrGeometryGetNumRegion	<filename> <geometry_index>
Integer	TdrGeometryGetNumState	<filename> <geometry_index>
List	TdrGeometryGetShift	<filename> <geometry_index>
Handle	TdrGeometryGetTagGroup	<filename> <geometry_index>
List	TdrGeometryGetTransform	<filename> <geometry_index>

A: Tcl Commands Interface of Sentaurus Data Explorer

State Commands

String	TdrGeometryGetType	<filename> <geometry_index>
Boolean	TdrGeometrySetName	<filename> <geometry_index> <name>
Boolean	TdrGeometrySetShift	<filename> <geometry_index> <shift_list>
Boolean	TdrGeometrySetTransform	<filename> <geometry_index> <transformation_list>

State Commands

Boolean	TdrStateDelete	<filename> <geometry_index> <state_index>
String	TdrStateGetName	<filename> <geometry_index> <state_index>
Handle	TdrStateGetTagGroup	<filename> <geometry_index> <state_index>
Boolean	TdrStateSetName	<filename> <geometry_index> <state_index> <name>

Region Commands

Integer	TdrRegionGetDimension	<filename> <geometry_index> <region_index>
String	TdrRegionGetMaterial	<filename> <geometry_index> <region_index>
String	TdrRegionGetName	<filename> <geometry_index> <region_index>
Integer	TdrRegionGetNumDataset	<filename> <geometry_index> <region_index> <state_index>
Handle	TdrRegionGetTagGroup	<filename> <geometry_index> <region_index>
String	TdrRegionGetType	<filename> <geometry_index> <region_index>
Boolean	TdrRegionSetMaterial	<filename> <geometry_index> <region_index> <material>
Boolean	TdrRegionSetName	<filename> <geometry_index> <region_index> <name>

Dataset Commands

Boolean	TdrDatasetDelete	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
Boolean	TdrDatasetDeleteByName	<filename> [<name>]
String	TdrDatasetGetLocation	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetName	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
Integer	TdrDatasetGetNumValue	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetQuantity	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetStructure	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
Handle	TdrDatasetGetTagGroup	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetType	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetUnit	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDatasetGetUnitLong	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
Boolean	TdrDatasetRename	<filename> [<old-name>] <new-name>
Boolean	TdrDatasetRenameQuantity	<filename> [<old-quantity>] <new-quantity>
Boolean	TdrDatasetSetName	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <name>
Boolean	TdrDatasetSetQuantity	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <quantity>

Data Value Commands

List	TdrDataGetAllCoordinates	<filename> <geometry_index> <region_index> <state_index> <dataset_index>
String	TdrDataGetComponent	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index> [<row>] [<col>]
Double	TdrDataGetCoordinate	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index> <coordinate_index>
Integer	TdrDataGetNumCol	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index>
Integer	TdrDataGetNumRow	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index>
String	TdrDataGetValue	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index>
Boolean	TdrDataSetComponent	<filename> <geometry_index> <region_index> <state_index> <dataset_index> <value_index> [<row>] [<col>] <value>

Tag Group Commands

Handle	TdrTagGroupCreate	<parent_tag_group> <name>
Boolean	TdrTagGroupDelete	<parent_tag_group> <tag_group_index>
Boolean	TdrTagGroupDeleteByName	<parent_tag_group> <tag_group_name>
Handle	TdrTagGroupGetByPath	<filename> [<geometry_index>] [<region_index>] [<state_index>] [<dataset_index>] <path>
String	TdrTagGroupGetName	<tag_group>
Integer	TdrTagGroupGetNumTag	<tag_group>
Integer	TdrTagGroupGetNumTagGroup	<tag_group>
Handle	TdrTagGroupGetTagGroup	<parent_tag_group> <tag_group_index>

Tag Commands

Boolean	TdrTagCreateScalar	<parent_tag_group> <name> <type> <value>
Boolean	TdrTagDelete	<tag_group> <tag_index>
Boolean	TdrTagDeleteByName	<tag_group> <tag_name>
String	TdrTagGetComponent	<tag_group> <tag_index> [<row>] [<col>]
String	TdrTagGetName	<tag_group> <tag_index>
Integer	TdrTagGetNumCol	<tag_group> <tag_index>
Integer	TdrTagGetNumRow	<tag_group> <tag_index>
String	TdrTagGetStructure	<tag_group> <tag_index>
String	TdrTagGetType	<tag_group> <tag_index>
String	TdrTagGetValue	<tag_group> <tag_index>
Boolean	TdrTagSetComponent	<tag_group> <tag_index> [<row>] [<col>] <value>

A: Tcl Commands Interface of Sentaurus Data Explorer
Tag Commands

APPENDIX B Structure of TDR Files

This appendix provides information about the structure of TDR files.

Overview

TDR is the standard data exchange format for TCAD Sentaurus tools. Its internal organization is displayed in [Figure 1](#). Any number of 1D, 2D, and 3D geometries of different types can be present in a TDR file. Each geometry is composed of regions and contains a set of states. A state contains a collection of datasets. Typically, states are used to store the simulation state at different points in time of a transient simulation. A dataset contains the values of a particular quantity for one region. As a consequence, a state typically contains multiple datasets for one quantity, because each dataset contains the values for only one specific region. The layout of the data values inside a dataset depends on the region and geometry types, and on the properties of the dataset.

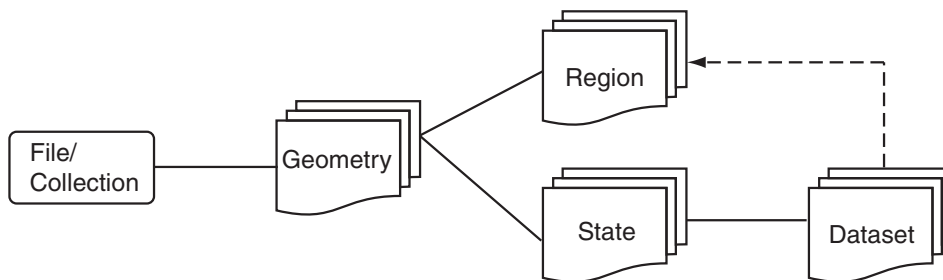


Figure 1 Basic structure of a TDR file

Tag Groups and Tags

Tags and tag groups allow you to associate arbitrary additional data to individual objects of a TDR file. This data is structured in a hierarchical way, similar to a file system. A tag group corresponds to a directory which contains tags and tag groups. Tags correspond to files in this analogy.

B: Structure of TDR Files

Tag Groups and Tags

A tag is a “name = value” pair. While the name is always a string, the value can be of different structure and value type. Available structure types are:

- Scalar
- Vector
- Matrix

Available value types are:

- Boolean
- 32-bit Integer
- 64-bit Integer
- Float
- Double
- Complex Float
- Complex Double
- String (currently restricted to scalar structure)

Tag groups can be associated with TDR objects of the following types:

- File (also called “Collection” in TDR terminology)
- Geometry
- Region
- State
- Dataset