



# Intel® Quartus® Prime Pro Edition User Guide

---

## Scripting

Updated for Intel® Quartus® Prime Design Suite: **21.3**



[Online Version](#)



[Send Feedback](#)

UG-20144

ID: **683432**

Version: **2021.10.04**

## Contents

---

<b>1. Command Line Scripting.....</b>	<b>5</b>
1.1. Benefits of Command-Line Executables.....	5
1.2. Command-Line Scripting Help.....	6
1.3. Project Settings with Command-Line Options.....	6
1.3.1. Option Precedence.....	7
1.4. Compilation with quartus_sh --flow.....	8
1.4.1. Resuming a Compilation with quartus_sh --flow.....	9
1.5. Text-Based Report Files.....	9
1.6. Using Command-Line Executables in Scripts.....	10
1.7. Common Scripting Examples.....	10
1.7.1. Create a Project and Apply Constraints.....	10
1.7.2. Check Design File Syntax.....	11
1.7.3. Create a Project and Synthesize a Netlist Using Netlist Optimizations.....	11
1.7.4. Archive and Restore Projects.....	12
1.7.5. Update Memory Contents Without Recompiling.....	12
1.7.6. Create Device Configuration Files.....	13
1.7.7. Fit a Design Using Multiple Seeds.....	14
1.8. The QFlow Script.....	14
1.8.1. --partition Option.....	15
1.9. Command-Line Scripting Revision History.....	16
<b>2. Tcl Scripting.....</b>	<b>18</b>
2.1. Tool Command Language.....	18
2.2. The Intel Quartus Prime Tcl Console Window.....	19
2.3. Intel Quartus Prime Tcl Packages.....	19
2.3.1. Loading Tcl Packages.....	21
2.3.2. Intel Quartus Prime Tcl API Help.....	21
2.4. Tcl Design Flow Controls.....	24
2.4.1. Creating Projects and Making Assignments.....	24
2.4.2. Compiling Designs.....	25
2.4.3. Reporting.....	25
2.4.4. Timing Analysis.....	27
2.5. Automating Script Execution.....	27
2.5.1. Execution Example.....	28
2.5.2. Controlling Processing.....	29
2.5.3. Displaying Messages.....	29
2.6. Other Scripting Features.....	29
2.6.1. Natural Bus Naming.....	29
2.6.2. Short Option Names.....	30
2.6.3. Collection Commands.....	30
2.6.4. Node Finder Commands.....	31
2.6.5. The get_names Command.....	38
2.6.6. The post_message Command.....	40
2.6.7. Accessing Command-Line Arguments.....	40
2.6.8. The quartus() Array.....	42
2.7. The Intel Quartus Prime Tcl Shell in Interactive Mode Example.....	42
2.8. The tclsh Shell.....	43

2.9. Tcl Scripting Basic Examples.....	43
2.9.1. Hello World Example.....	43
2.9.2. Variables.....	44
2.9.3. Substitutions.....	44
2.9.4. Arithmetic.....	45
2.9.5. Lists.....	45
2.9.6. Arrays.....	45
2.9.7. Control Structures.....	46
2.9.8. Procedures.....	47
2.9.9. File I/O.....	47
2.9.10. Syntax and Comments.....	48
2.9.11. External References.....	49
2.10. Tcl Scripting Revision History.....	49
<b>3. TCL Commands and Packages.....</b>	<b>51</b>
3.1. TCL Commands and Packages Summary.....	51
3.1.1. ::quartus::backannotate.....	66
3.1.2. ::quartus::bpps.....	69
3.1.3. ::quartus::chip_planner.....	90
3.1.4. ::quartus::design.....	98
3.1.5. ::quartus::device.....	110
3.1.6. ::quartus::drc.....	114
3.1.7. ::quartus::eco.....	127
3.1.8. ::quartus::external_memif_toolkit.....	141
3.1.9. ::quartus::fifo.....	162
3.1.10. ::quartus::flng.....	169
3.1.11. ::quartus::flow.....	178
3.1.12. help.....	0
3.1.13. ::quartus::insystem_memory_edit.....	183
3.1.14. ::quartus::insystem_source_probe.....	190
3.1.15. ::quartus::interactive_synthesis.....	195
3.1.16. ::quartus::ipgen.....	203
3.1.17. ::quartus::iptclgen.....	206
3.1.18. ::quartus::jtag.....	209
3.1.19. ::quartus::logic_analyzer_interface.....	221
3.1.20. ::quartus::misc.....	227
3.1.21. ::quartus::names.....	238
3.1.22. ::quartus::periph.....	240
3.1.23. ::quartus::pfg.....	256
3.1.24. ::quartus::project.....	257
3.1.25. ::quartus::qshm.....	323
3.1.26. ::quartus::report.....	327
3.1.27. ::quartus::sdc.....	346
3.1.28. ::quartus::sdc_ext.....	375
3.1.29. ::quartus::sta.....	393
3.1.30. ::quartus::stp.....	491
3.1.31. ::quartus::tdc.....	496
3.1.32. ::quartus::uno.....	498
3.2. Tcl Commands and Packages Revision History.....	499
<b>4. Intel Quartus Prime Pro Edition User Guide Scripting Archives.....</b>	<b>500</b>

<b>A. Intel Quartus Prime Pro Edition User Guides.....</b>	<b>501</b>
--	------------

## 1. Command Line Scripting

---

FPGA design software that easily integrates into your design flow saves time and improves productivity. The Intel® Quartus® Prime software provides you with a command-line executable for each step of the FPGA design flow to make the design process customizable and flexible.

The command-line executables are completely interchangeable with the Intel Quartus Prime GUI, allowing you to use the exact combination of tools that best suits your needs.

### Related Information

- [Tcl Design Examples](#)
- [TCL Commands and Packages](#) on page 51

### 1.1. Benefits of Command-Line Executables

Intel Quartus Prime command-line executables give you precise control over each step of the design flow, reduce memory requirements, and improve performance.

You can group Intel Quartus Prime executable files into a script, batch file, or a makefile to automate design flows. These scripting capabilities facilitate the integration of Intel Quartus Prime software and other EDA synthesis, simulation, and verification software. Automatic design flows can perform on multiple computers simultaneously and easily archive and restore projects.

Command-line executables add flexibility without sacrificing the ease-of-use of the Intel Quartus Prime GUI. You can use the Intel Quartus Prime GUI and command-line executables at different stages in the design flow. For example, you might use the Intel Quartus Prime GUI to edit the floorplan for the design, use the command-line executables to perform place-and-route, and return to the Intel Quartus Prime GUI to perform debugging.

Command-line executables reduce the amount of memory required during each step in the design flow. Since each executable targets only one step in the design flow, the executables themselves are relatively compact, both in file size and the amount of memory used during processing. This memory usage reduction improves performance, and is particularly beneficial in design environments where heavy usage of computing resources results in reduced memory availability.

### Related Information

- [About Command-Line Executables](#)  
in Intel Quartus Prime Help

## 1.2. Command-Line Scripting Help

Help for command-line executables is available through different methods. You can access help built into the executables with command-line options. You can use the Intel Quartus Prime Command-Line and Tcl API Help browser for an easy graphical view of the help information.

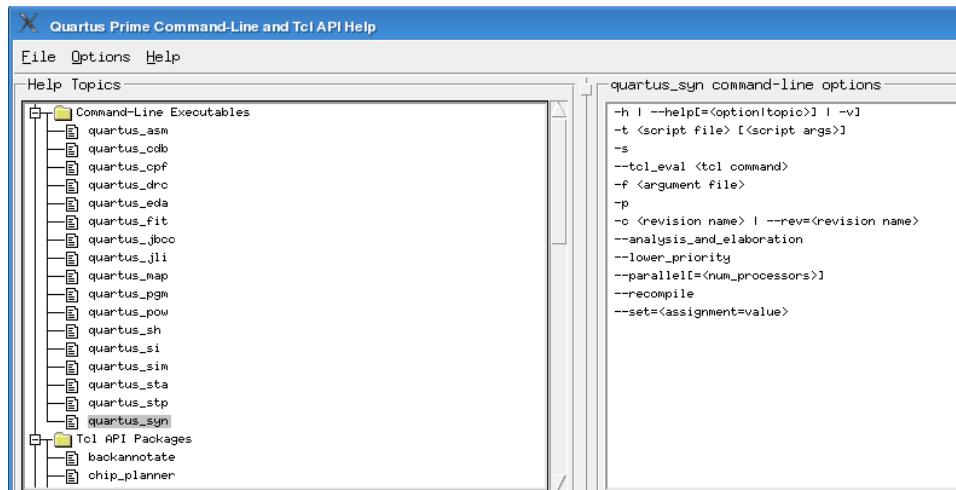
To use the Intel Quartus Prime Command-Line and Tcl API Help browser, type the following command:

```
quartus_sh --qhelp
```

This command starts the Intel Quartus Prime Command-Line and Tcl API Help browser, a viewer for information about the Intel Quartus Prime Command-Line executables and Tcl API.

Use the `-h` option with any of the Intel Quartus Prime Command-Line executables to get a description and list of supported options. Use the `--help=<option name>` option for detailed information about each option.

**Figure 1.** Intel Quartus Prime Command-Line and Tcl API Help Browser



## 1.3. Project Settings with Command-Line Options

The Intel Quartus Prime software command-line executables accept arguments to set project variables and access common settings.

To make assignments to an individual entity you can use the Intel Quartus Prime Tcl scripting API. On existing projects, you can also open the project in the Intel Quartus Prime GUI, change the assignment, and close the project. The changed assignment is updated in the `.qsf`. Any command-line executables that are run after this update use the updated assignment.

### Related Information

- [Compilation with quartus\\_sh --flow](#) on page 8
- [Intel Quartus Prime Settings File \(.qsf\) Definition](#)  
in Intel Quartus Prime Help

- Intel Quartus Prime Pro Edition Settings File Reference Manual

### 1.3.1. Option Precedence

Project assignments follow a set of precedence rules. Assignments for a project can exist in three places:

- Intel Quartus Prime Settings File (.qsf)
- The compiler database
- Command-line options

The .qsf file contains all the project-wide and entity-level assignments and settings for the current revision for the project. The compiler database contains the result of the last compilation in the /db directory, and reflects the assignments at the moment when the project was compiled. Updated assignments first appear in the compiler database and later in the .qsf file.

Command-line options override any conflicting assignments in the .qsf file or the compiler database files. To specify whether the .qsf or compiler database files take precedence for any assignments not specified in the command-line, use the option --read\_settings\_files.

**Table 1. Precedence for Reading Assignments**

Option Specified	Precedence for Reading Assignments
--read_settings_files = on (Default)	1. Command-line options 2. The .qsf for the project 3. Project database (db directory, if it exists) 4. Intel Quartus Prime software defaults
--read_settings_files = off	1. Command-line options 2. Project database (db directory, if it exists) 3. Intel Quartus Prime software defaults

The --write\_settings\_files command-line option lists the locations to which assignments are written..

**Table 2. Location for Writing Assignments**

Option Specified	Location for Writing Assignments
--write_settings_files = on (Default)	.qsf file and compiler database
--write_settings_files = off	Compiler database

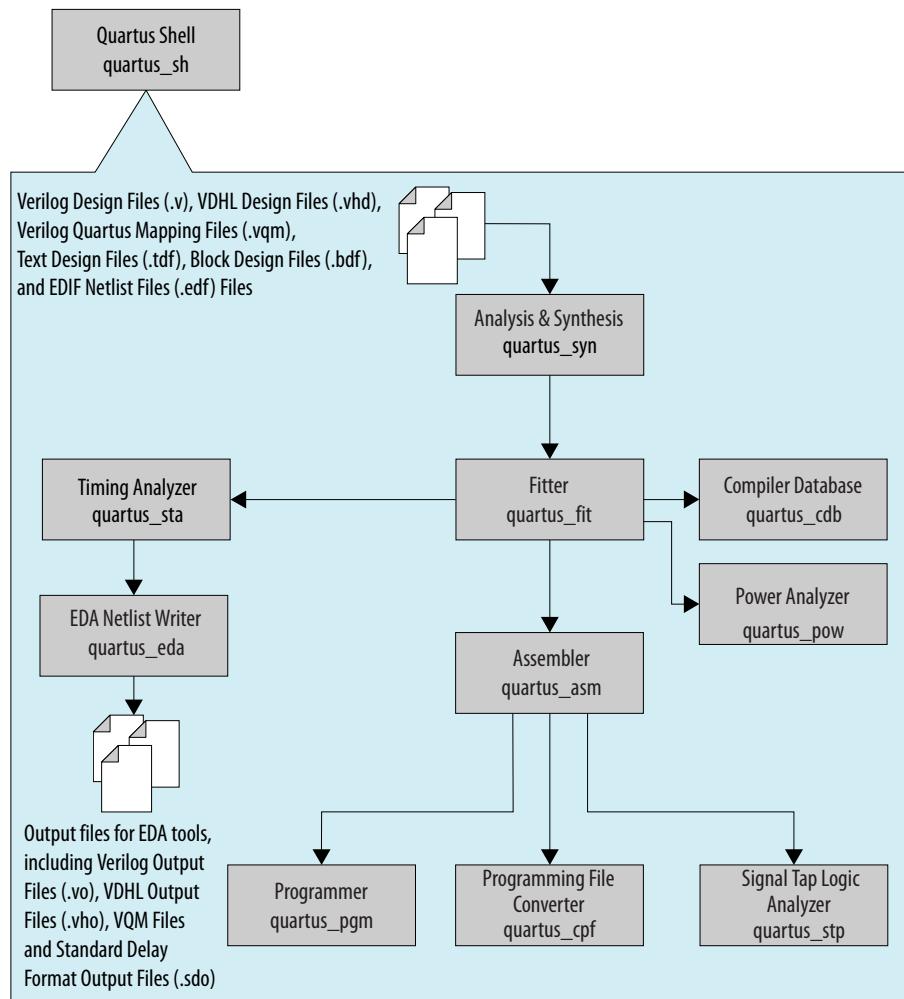
Any assignment not specified as a command-line option or found in the .qsf file or compiler database file is set to its default value.

Use the options --read\_settings\_files=off and --write\_settings\_files=off (where appropriate) to optimize the way that the Intel Quartus Prime software reads and updates settings files.

## 1.4. Compilation with quartus\_sh --flow

The figure shows a typical Intel Quartus Prime FPGA design flow using command-line executables.

**Figure 2. Typical Design Flow**



Use the `quartus_sh` executable with the `--flow` option to perform a complete compilation flow with a single command. The `--flow` option supports the smart recompile feature and efficiently sets command-line arguments for each executable in the flow.

You can resume an interrupted compilation with the `-resume` argument of the `--flow` option.

The following example runs analysis & synthesis, fitter, timing analysis, and programming file generation with a single command:

```
quartus_sh --flow compile filtref
```

**Tip:** For information about specialized flows, type `quartus_sh --help=flow` at a command prompt.

#### **Related Information**

[Resuming a Compilation with `quartus\_sh --flow`](#) on page 9

### **1.4.1. Resuming a Compilation with `quartus_sh --flow`**

You can resume a compilation flow for a project from the last valid step completed in the flow with the `-resume` option of the `quartus_sh --flow` command. If you want to resume a compilation flow, ensure that no settings that affect the subsequent compilation stages have changed from initial start of the compilation.

You can also use the `-start` and `-stop` options of `quartus_sh --flow` command to start and stop a compilation flow at specific compilation tasks.

Resuming a compilation flow also updates the Compilation Dashboard to show how the flow is progressing.

For command syntax and example of how to use the flow resume feature, run `quartus_sh --help=flow` at a command prompt.

#### **Related Information**

[Using the Compilation Dashboard](#)

### **1.5. Text-Based Report Files**

Each command-line executable creates a text report file when it is run. These files report success or failure, and contain information about the processing performed by the executable.

Report file names contain the revision name and the short-form name of the executable that generated the report file, in the format `<revision>.<executable>.rpt`. For example, using the `quartus_fit` executable to place and route a project with the revision name **design\_top** generates a report file named `design_top.fit.rpt`. Similarly, using the `quartus_sta` executable to perform timing analysis on a project with the revision name **fir\_filter** generates a report file named `fir_filter.sta.rpt`.

As an alternative to parsing text-based report files, you can use the `::quartus::report` Tcl package.

#### **Related Information**

- [Text-Format Report File \(.rpt\) Definition](#)  
in Intel Quartus Prime Help
- [::quartus::report](#)  
in Intel Quartus Prime Help

## 1.6. Using Command-Line Executables in Scripts

You can use command-line executables in scripts that control other software, in addition to Intel Quartus Prime software. For example, if your design flow uses third-party synthesis or simulation software, and you can run this other software at the command prompt, you can group those commands with Intel Quartus Prime executables in a single script.

To set up a new project and apply individual constraints, such as pin location assignments and timing requirements, you must use a Tcl script or the Intel Quartus Prime GUI.

Command-line executables are very useful for working with existing projects, for making common global settings, and for performing common operations. For more flexibility in a flow, use a Tcl script. Additionally, Tcl scripts simplify passing data between different stages of the design flow.

For example, you can create a UNIX shell script to run a third-party synthesis software, place-and-route the design in the Intel Quartus Prime software, and generate output netlists for other simulation software.

## 1.7. Common Scripting Examples

You can create scripts including command line executable to control common Intel Quartus Prime processes.

### 1.7.1. Create a Project and Apply Constraints

The command-line executables include options for common global project settings and commands. You can use a Tcl script to apply constraints such as pin locations and timing assignments. You can write a Tcl constraint file, or generate one for an existing project by clicking **Project > Generate Tcl File for Project**.

The example creates a project with a Tcl script and applies project constraints using the tutorial design files in the <Intel Quartus Prime *installation directory*

```
project_new filtref -overwrite
# Assign family, device, and top-level file
set_global_assignment -name FAMILY "Arria 10"
set_global_assignment -name DEVICE <Device>
set_global_assignment -name VERILOG_FILE filtref.v
# Assign pins
set_location_assignment -to clk Pin_28
set_location_assignment -to clkx2 Pin_29
set_location_assignment -to d[0] Pin_139
set_location_assignment -to d[1] Pin_140
#
project_close
```

Save the script in a file called `setup_proj.tcl` and type the commands illustrated in the example at a command prompt to create the design, apply constraints, compile the design, and perform fast-corner and slow-corner timing analysis. Timing analysis results are saved in two files, `filtref_sta_1.rpt` and `filtref_sta_2.rpt`.

```
quartus_sh -t setup_proj.tcl
quartus_syn filtref
quartus_fit filtref
```

```
quartus_asm filtref
quartus_sta filtref --model=fast --export_settings=off
mv filtref_sta.rpt filtref_sta_1.rpt
quartus_sta filtref --export_settings=off
mv filtref_sta.rpt filtref_sta_2.rpt
```

Type the following commands to create the design, apply constraints, and compile the design, without performing timing analysis:

```
quartus_sh -t setup_proj.tcl
quartus_sh --flow compile filtref
```

The `quartus_sh --flow compile` command performs a full compilation, and is equivalent to clicking the **Start Compilation** button in the toolbar.

### 1.7.2. Check Design File Syntax

The .tcl script example below assumes the Intel Quartus Prime software **fir\_filter** tutorial project exists in the current directory. You can find the **fir\_filter** project in the `<Intel Quartus Prime directory>/qdesigns/fir_filter` directory unless the Intel Quartus Prime software tutorial files are not installed.

When options are not specified, the executable uses the project database values. If not specified in the project database, the executable uses the Intel Quartus Prime software default values.

To run this script, save this script to a file such as `check_syntax.tcl` and then run the following command from a command prompt: `quartus_syn -t check_syntax.tcl`.

```
set dir [pwd]; # set dir to current working directory
# assign quartus_files variable to all files within current working directory
# asterisk (*) may be changed to specific file extensions (i.e. *.v, *.vhdl,
*.etc)
set quartus_files [glob -directory $dir *]
# open project fir_filter with revision name filtref
project_open fir_filter -revision filtref
foreach file $quartus_files {
    post_message $file;           # echo which file was analyzed
    analyze_files -files $file -library work; # analyze file for syntax
}
project_close; # close project
```

### 1.7.3. Create a Project and Synthesize a Netlist Using Netlist Optimizations

This example creates a new Intel Quartus Prime project with a file `top.edf` as the top-level entity. The `--enable_register_retimining=on` and `--enable_wysiwyg_resynthesis=on` options cause `quartus_map` to optimize the design using gate-level register retiming and technology remapping.

The --part option causes quartus\_syn to target a device. To create the project and synthesize it using the netlist optimizations described above, type the command shown in this example at a command prompt.

```
quartus_syn top --source=top.edf --enable_register_retimining=on  
--enable_wysiwyg_resynthesis=on --part=<part>
```

#### 1.7.4. Archive and Restore Projects

You can archive or restore an Intel Quartus Prime Archive File (.qar) with a single command. This makes it easy to take snapshots of projects when you use batch files or shell scripts for compilation and project management.

Use the --archive or --restore options for quartus\_sh as appropriate. Type the command shown in the example at a command prompt to archive your project.

```
quartus_sh --archive <project name>
```

The archive file is automatically named <project name>.qar. If you want to use a different name, type the command with the -output option as shown in example the example.

```
quartus_sh --archive <project name> -output <filename>
```

To restore a project archive, type the command shown in the example at a command prompt.

```
quartus_sh --restore <archive name>
```

The command restores the project archive to the current directory and overwrites existing files.

##### Related Information

[Managing Intel Quartus Prime Projects](#)

#### 1.7.5. Update Memory Contents Without Recompiling

You can use two commands to update the contents of memory blocks in your design without recompiling. Use the quartus\_cdb executable with the --update\_mif option to update memory contents from .mif or .hexout files. Then, rerun the assembler with the quartus\_asm executable to regenerate the .sof, .pof, and any other programming files.

```
quartus_cdb --update_mif <project name> [--rev=<revision name>]  
quartus_asm <project name> [--rev=<revision name>]
```

The example shows the commands for a DOS batch file for this example. With a DOS batch file, you can specify the project name and the revision name once for both commands. To create the DOS batch file, paste the following lines into a file called update\_memory.bat.

```
quartus_cdb --update_mif %1 --rev=%2  
quartus_asm %1 --rev=%2
```

To run the batch file, type the following command at a command prompt:

```
update_memory.bat <project name> <revision name>
```

## 1.7.6. Create Device Configuration Files

You can use the `quartus_cpf` or `quartus_pfg` command line executables to generate different types of device configuration files at the command line, depending on your target device.

- `quartus_pfg`—controls the same programming file generation functions as the **Programming File Generator** dialog box in the Intel Quartus Prime software GUI, and supports programming file generation for Intel Stratix® 10 and Intel Agilex™ device families.

**Table 3. quartus\_pfg Command Examples**

Command Function	Command Syntax
Specify the ASX4 operation mode, convert .sof to .jic	<code>quartus_pfg -c -o device=MT25QU512 -o mode=ASX4 -o flash_loader=1SG280HN3S3 \ project.sof project.jic</code>
Access full command-line syntax help	<code>quartus_pfg --help</code>

- `quartus_cpf`—controls the same functions as the **Convert Programming Files** dialog box in the Intel Quartus Prime software GUI, and supports programming file generation for all device families prior to the Intel Stratix 10 device family.

**Table 4. quartus\_cpf Command Examples**

Command Function	Command Syntax
Create an option file that turns on compression, type the following command at a command prompt	<code>quartus_cpf -w &lt;filename&gt;.opt</code>
Create a compressed .pof that targets an EPICS64 device	<code>quartus_cpf --convert --option=&lt;filename&gt;.opt --device=EPICS64 \ &lt;file&gt;.sof &lt;file&gt;.pof</code>
Save configuration options in a conversion setup file (.cof)	<code>quartus_cpf --convert &lt;file&gt;.cof</code>
Convert a .sof programming file to CvP periphery image (*.jam) file	<code>quartus_cpf -c &lt;filename&gt;.sof &lt;filename&gt;.jam --cvp</code>
Access full command-line syntax help	<code>quartus_cpf --help</code>

**Note:** For complete Intel Quartus Prime command line executable syntax and examples, refer to [Command-Line Scripting Help](#) on page 6.

### 1.7.7. Fit a Design Using Multiple Seeds

This shell script example assumes that the Intel Quartus Prime software tutorial project called **fir\_filter** exists in the current directory (defined in the file **fir\_filter.qpf**). If the tutorial files are installed on your system, this project exists in the *<Intel Quartus Prime directory>/qdesigns<quartus\_version\_number>/fir\_filter* directory.

Because the top-level entity in the project does not have the same name as the project, you must specify the revision name for the top-level entity with the **--rev** option. The **--seed** option specifies the seeds to use for fitting.

A seed is a parameter that affects the random initial placement of the Intel Quartus Prime Fitter. Varying the seed can result in better performance for some designs.

After each fitting attempt, the script creates new directories for the results of each fitting attempt and copies the complete project to the new directory so that the results are available for viewing and debugging after the script has completed.

```
#!/bin/sh
ERROR_SEEDS=""
quartus_syn fir_filter --rev=filtref
# Iterate over a number of seeds
for seed in 1 2 3 4 5
do
echo "Starting fit with seed=$seed"
# Perform a fitting attempt with the specified seed
quartus_fit fir_filter --seed=$seed --rev=filtref
# If the exit-code is non-zero, the fitting attempt was
# successful, so copy the project to a new directory
if [ $? -eq 0 ]
then
mkdir ..../fir_filter-seed_$seed
mkdir ..../fir_filter-seed_$seed/db
cp * ..../fir_filter-seed_$seed
cp db/* ..../fir_filter-seed_$seed/db
else
ERROR_SEEDS="$ERROR_SEEDS $seed"
fi
done
if [ -z "$ERROR_SEEDS" ]
then
echo "Seed sweeping was successful"
exit 0
else
echo "There were errors with the following seed(s)"
echo $ERROR_SEEDS
exit 1
fi
```

*Tip:*

Use Design Space Explorer II (DSE) included with the Intel Quartus Prime software script (by typing **quartus\_dse** at a command prompt) to improve design performance by performing automated seed sweeping.

## 1.8. The QFlow Script

A Tcl/Tk-based graphical interface called QFlow is included with the command-line executables. You can use the QFlow interface to open projects, launch some of the command-line executables, view report files, and make some global project assignments.

The QFlow interface can run the following command-line executables:

- quartus\_syn (Analysis and Synthesis)
- quartus\_fit (Fitter)
- quartus\_sta (Timing Analyzer)
- quartus\_asm (Assembler)
- quartus\_eda (EDA Netlist Writer)

To view floorplans or perform other GUI-intensive tasks, launch the Intel Quartus Prime software.

Start QFlow by typing the following command at a command prompt:

```
quartus_sh -g
```

*Tip:* The QFlow script is located in the <Intel Quartus Prime directory> / common/tcl/apps/qflow/ directory.

### 1.8.1. --partition Option

The --partition option is for the --simulation top-level argument for quartus\_eda. This option selects an individual partition as the netlist output.

#### --exclude\_sub\_partitions

The --exclude\_sub\_partitions flag limits the output to the netlist of this partition only. This flag is only valid when you use the --partition option, this flag outputs the netlist belonging to the partition specified. The software instantiates sub-partitions as separate modules.

For no partition argument, the software writes the entire design out to a single file. The partition argument takes a name of a partition in the design. The sub-option is a flag only and takes no arguments.

When you specify the --exclude\_sub\_partitions flag, the software only writes out the contents of the selected partition. Sub-partitions are instantiated as separate modules. Each call of quartus\_eda writes one netlist. If you write out the design one partition at a time, excluding sub partitions, they need to call quartus\_eda for each partition in the design including the root.

#### root\_partition

You can specify the root\_partition flag to get the full design. You can provide the partition option to write to a netlist file (.vo or .vho file). The file contains all the logic and atoms corresponding the contents of the specified partition along with its sub-partition.

#### --rename

Additionally, you have the option to rename a module name in the generated netlist file using the --rename option. By default, the software uses the partition name as the module name in the netlist file. This option is only valid when you use the partition option. You can elect to rename any module using --module\_name=abc=xyz --module\_name=def=prq. The generated file names format is: <revision>.<module>

*or partition name>.<vo or vho>* By default, the software writes the netlist file to the simulation/<3rd party simulation tool> directory (eg simulation/modelsim), unless you specify an output\_directory (using a command line option or .qsf assignment).

## 1.9. Command-Line Scripting Revision History

The following revision history applies to this chapter:

**Table 5. Document Revision History**

Document Version	Intel Quartus Prime Version	Changes
2021.03.29	21.1	<ul style="list-style-type: none"> <li>Added "Resuming a Compilation with quartus_sh --flow"</li> <li>Revised "Check Design File Syntax".</li> </ul>
2020.12.14	20.4	<ul style="list-style-type: none"> <li>Updated "Create Device Configuration Files" to include references to the cvp command option and the quartus_pfg command.</li> </ul>
2020.04.13	20.1	Added --partition option.
2017.05.08	17.0.0	<ul style="list-style-type: none"> <li>Reorganized content on topics: Benefits of Command-Line Executables and Project Settings with Command-Line Options.</li> <li>Removed mentions to unsupported executables and options.</li> <li>Removed topics: Introductory Example and Common Scripting Examples</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>Implemented Intel rebranding.</li> </ul>
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
2015.05.04	15.0.0	Remove descriptions of makefile support that was removed from software in 14.0.
December 2014	14.1.0	Updated DSE II commands.
June 2014	14.0.0	Updated formatting.
November 2013	13.1.0	Removed information about -silnet qmegawiz command
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	Corrected quartus_qpf example usage. Updated examples.
December 2010	10.1.0	Template update. Added section on using a script to regenerate megafunction variations. Removed references to the Classic Timing Analyzer (quartus_tan). Removed Qflow illustration.
July 2010	10.0.0	Updated script examples to use quartus_sta instead of quartus_tan, and other minor updates throughout document.
November 2009	9.1.0	Updated Table 2-1 to add quartus_jli and quartus_jbcc executables and descriptions, and other minor updates throughout document.
March 2009	9.0.0	No change to content.
November 2008	8.1.0	Added the following sections:  <i>continued...</i>

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"><li>• "The MegaWizard Plug-In Manager" on page 2-11</li><li>• "Command-Line Support" on page 2-12</li><li>• "Module and Wizard Names" on page 2-13</li><li>• "Ports and Parameters" on page 2-14</li><li>• "Invalid Configurations" on page 2-15</li><li>• "Strategies to Determine Port and Parameter Values" on page 2-15</li><li>• "Optional Files" on page 2-15</li><li>• "Parameter File" on page 2-16</li><li>• "Working Directory" on page 2-17</li><li>• "Variation File Name" on page 2-17</li><li>• "Create a Compressed Configuration File" on page 2-21</li><li>• Updated "Option Precedence" on page 2-5 to clarify how to control precedence</li><li>• Corrected Example 2-5 on page 2-8</li><li>• Changed Example 2-1, Example 2-2, Example 2-4, and Example 2-7 to use the EP1C12F256C6 device</li><li>• Minor editorial updates</li><li>• Updated entire chapter using 8½" x 11" chapter template</li></ul>
May 2008	8.0.0	<ul style="list-style-type: none"><li>• Updated "Referenced Documents" on page 2-20.</li><li>• Updated references in document.</li></ul>

## 2. Tcl Scripting

---

You can use Tcl scripts, as an alternative to the GUI, to control the function and operation of Intel Quartus Prime software.

For example, you can use Tcl scripts to perform the following tasks:

- Manage Intel Quartus Prime projects
- Specify assignments and constraints
- Compile your design
- Perform timing analysis
- Generate and view reports about your project

You can also use Tcl scripts to migrate a project or project settings. For example, when working with different projects targeting the same prototype or development board, you can define a Tcl script to automate pin assignments for each project, rather than entering the assignments individually in the GUI. You can automatically generate a Tcl script based on current project assignments, which simplifies transferring the assignments to another project.

The Intel Quartus Prime software Tcl commands follow familiar EDA industry Tcl application programming interface (API) standards for command-line options. If you encounter an error with a command argument, the Tcl interpreter includes help information showing correct usage.

This chapter includes sample Tcl scripts for automating tasks in the Intel Quartus Prime software, along with a complete reference of all supported Tcl commands and arguments. You can modify the example scripts for use with your own designs. Refer to Design Examples section of the Support area on the Intel website.

### Related Information

- [Tcl Design Examples](#)
- [TCL Commands and Packages](#) on page 51

### 2.1. Tool Command Language

Tcl (pronounced “tickle”) stands for Tool Command Language, and is the industry-standard scripting language. Tcl supports control structures, variables, network socket access, and APIs.

With Tcl, you can work seamlessly across most development platforms. Synopsys\*, Mentor Graphics\*, and Intel software products support the Tcl language.

By combining Tcl commands and Intel Quartus Prime API functions, you can create your own procedures and automate your design flow. Run Intel Quartus Prime software in batch mode, or execute individual Tcl commands interactively in the Intel Quartus Prime Tcl shell.

Intel Quartus Prime software supports Tcl/Tk version 8.5, supplied by the Tcl DeveloperXchange.

## 2.2. The Intel Quartus Prime Tcl Console Window

A Tcl Console Window is available in the Intel Quartus Prime software GUI by clicking **View > Tcl Console**. You can run Intel Quartus Prime Tcl commands directly in the **Tcl Console** window. the Intel Quartus Prime Tcl shell interprets all Tcl commands that you type in the **Tcl Console**.

**Note:** Some shell commands such as `cd`, `ls`, and others can be run in the Tcl Console window, with the `Tcl exec` command. However, for best results, run shell commands and Intel Quartus Prime executables from a system command prompt outside of the Intel Quartus Prime software GUI.

Tcl messages appear in the **System** tab (**Messages** window). Errors and messages written to `stdout` and `stderr` also are shown in the Intel Quartus Prime **Tcl Console** window.

## 2.3. Intel Quartus Prime Tcl Packages

The Intel Quartus Prime software groups Tcl commands into packages by function.

**Note:** Refer to [TCL Commands and Packages](#) on page 51 for a comprehensive reference of all Intel Quartus Prime Tcl packages and commands.

**Table 6. Intel Quartus Prime Tcl Packages**

Package Name	Package Description
backannotate	Back-annotate the Compiler's assignments.
bpps	Floorplan IP interfaces and other device resources in Interface Planner.
chip_planner	Identify and modify resource usage and routing with the Chip Planner.
design	Manipulate project databases, including the assignments database, to enable the creation of instance assignments without modifying the .qsf file.
device	Get device and family information from the device database.
dni_sdc	Set false path, input delay, or output delay SDC constraints.
drc	Interact with Design Assistant design rule checks.
eco	Specify engineering change orders after design compilation.
external_memif_toolkit	Interact with external memory interfaces and debug components.
fif	Contains the set of Tcl functions for using the Fault Injection File (FIF) Driver
flng	Query properties of generic objects.
flow	Compile a project, run command-line executables, and other compilation flows.

*continued...*

Package Name	Package Description
help	Tcl command help.
insystem_memory_edit	Read and edit memory contents in Intel FPGA devices.
insystem_source_probe	Interact with the In-System Sources and Probes tool in an Intel device.
interactive_synthesis	Interactive synthesis controls.
ipgen	IP generation controls.
iptclgen	Memory IP generation controls.
jtag	Control the JTAG chain.
logic_analyzer_interface	Query and modify the Logic Analyzer Interface output pin state.
misc	Perform miscellaneous tasks such as enabling natural bus naming, package loading, and message posting.
names	Gets or sets assignment names.
periph	Interact with the interface pins.
pfg	Controls the Programming File Generator.
project	Create and manage projects and revisions, make any project assignments including timing assignments.
project_ui	Query the GUI.
qshm	Client and server controls.
report	Get information from report tables, create custom reports.
rtl	Traverse and query the RTL netlist of your design.
sdc	Specify constraints and exceptions to the Timing Analyzer.
sdc_ext	Intel FPGA-specific SDC commands.
sta	Contains the set of Tcl functions for obtaining advanced information from the Timing Analyzer.
stp	Run the Signal Tap Logic Analyzer.
tdc	Obtain information from the Timing Analyzer.

To keep memory requirements as low as possible, only the minimum number of packages load automatically with each Intel Quartus Prime executable. To run commands from other packages, load those packages beforehand.

Run your scripts with executables that include the packages you use in the scripts. For example, to use commands in the sdc\_ext package, you must use the quartus\_sta executable because quartus\_sta is the only executable with support for the sdc\_ext package.

The following command prints lists of the packages loaded or available to load for an executable, to the console:

```
<executable name> --tcl_eval help
```

For example, type the following command to list the packages loaded or available to load by the quartus\_fit executable:

```
quartus_fit --tcl_eval help
```

#### Related Information

- [Tcl Design Examples](#)
- [TCL Commands and Packages](#) on page 51

### 2.3.1. Loading Tcl Packages

To load an Intel Quartus Prime Tcl package, use the `load_package` command as follows:

```
load_package [-version <version number>] <package name>
```

This command is similar to `package require`, but it allows to alternate between different versions of an Intel Quartus Prime Tcl package.

### 2.3.2. Intel Quartus Prime Tcl API Help

Intel Quartus Prime Tcl help allows easy access to information about the Intel Quartus Prime Tcl commands.

- This command opens the Intel Quartus Prime Command-Line and Tcl API help browser, which documents all commands and options in the Intel Quartus Prime Tcl API. At a system command prompt, access the Intel Quartus Prime Tcl API Help by typing:

```
quartus_sh --qhelp
```

- The Tcl API Help can be accessed from the Tcl console as well. At a Tcl prompt, type

```
help
```

to access the help information. The output is:

The Tcl console provides help options that display specific information:

**Table 7. Help Options Available in the Intel Quartus Prime Tcl Environment**

Help Command	Description
<code>help</code>	Displays complete list of available Intel Quartus Prime Tcl packages.
<code>help -tcl</code>	Explains how to load Tcl packages and access command-line help.
<code>help -pkg &lt;package_name&gt; [-version &lt;version number&gt;]</code>	Displays help commands of the Intel Quartus Prime package that you specify, including the list of available Tcl commands. <ul style="list-style-type: none"><li>• If you do not specify <code>-version</code>, the Intel Quartus Prime software loads the latest version of the package.</li><li>• If the package is not loaded, the Intel Quartus Prime software displays the help for the latest version of the package.</li></ul>

*continued...*

Help Command	Description
	<p>Examples:</p> <pre>help -pkg ::quartus::project help -pkg project help -pkg project -version 1.0</pre>
<code>&lt;command_name&gt; -h</code> or <code>&lt;command_name&gt; -help</code>	<p>Displays the short help of a Intel Quartus Prime Tcl command in a loaded package. Examples:</p> <pre>project_open -h project_open -help</pre>
<code>package require ::quartus::&lt;package name&gt;[&lt;version&gt;]</code>	<p>Loads a specific version of an Intel Quartus Prime Tcl package. If you do not specify -version, the Intel Quartus Prime software loads the latest version of the package.</p> <p>Example:</p> <pre>package require ::quartus::project 1.0</pre> <p>This command is similar to the <code>load_package</code> command</p>
<code>load_package &lt;package name&gt; [-version &lt;version number&gt;]</code>	<p>Allows you to alternate between different versions of the same package.</p> <p>Example:</p> <pre>load_package ::quartus::project -version 1.0</pre>
<code>help -cmd &lt;command_name&gt; [-version &lt;version&gt;]</code> or <code>&lt;command_name&gt; -long_help</code>	<p>Displays the complete help text for an Intel Quartus Prime Tcl command. If you do not specify -version, the Intel Quartus Prime software loads the latest version of the package.</p> <p>Examples:</p> <pre>project_open -long_help help -cmd project_open help -cmd project_open -version 1.0</pre>
<code>help -examples</code>	<p>Displays examples of Intel Quartus Prime Tcl usage.</p>
<code>help -quartus</code>	<p>To view help on the predefined global Tcl array that contains project information and information about the Intel Quartus Prime executable that is currently running.</p>
<code>quartus_sh --qhelp</code>	<p>Launches the Tk viewer for Intel Quartus Prime command-line help and display help for the command-line executables and Tcl API packages.</p>
<code>help -timequestinfo</code>	<p>To view help on the predefined global "TimeQuestInfo" Tcl array that contains delay model information and speed grade information of a Timing Analyzer design that is currently running.</p>

The Tcl API help is also available in Intel Quartus Prime online help. Search for the command or package name to find details about that command or package.

### 2.3.2.1. Command-Line Options

You can use any of the following command line options with executables that support Tcl:

**Table 8. Command-Line Options Supporting Tcl Scripting**

Command-Line Option	Description
--script=<script file> [<script args>]	Run the specified Tcl script with optional arguments.
-t <script file> [<script args>]	Run the specified Tcl script with optional arguments. The -t option is the short form of the --script option.
--shell	Open the executable in the interactive Tcl shell mode.
-s	Open the executable in the interactive Tcl shell mode. The -s option is the short form of the --shell option.
--tcl_eval <tcl command>	Evaluate the remaining command-line arguments as Tcl commands. For example, the following command displays help for the project package: <code>quartus_sh --tcl_eval help -pkg project</code>

#### 2.3.2.1.1. Run a Tcl Script

Running an executable with the -t option runs the specified Tcl script. You can also specify arguments to the script. Access the arguments through the `argv` variable, or use a package such as `cmdline`, which supports arguments of the following form:

```
-<argument name> <argument value>
```

The `cmdline` package is included in the `<Intel Quartus Prime directory>/common/tcl/packages` directory.

For example, to run a script called `myscript.tcl` with one argument, `Stratix`, type the following command at a system command prompt:

```
quartus_sh -t myscript.tcl Stratix
```

#### 2.3.2.1.2. Interactive Shell Mode

Running an executable with the -s option starts an interactive Tcl shell. For example, to open the Intel Quartus Prime Timing Analyzer executable in interactive shell mode, type:

```
quartus_sta -s
```

Commands you type in the Tcl shell are interpreted when you press Enter. To run a Tcl script in the interactive shell type:

```
source <script name>
```

If a command is not recognized by the shell, it is assumed to be external and executed with the `exec` command.

### 2.3.2.1.3. Evaluate as Tcl

Running an executable with the `--tcl_eval` option causes the executable to immediately evaluate the remaining command-line arguments as Tcl commands. This can be useful if you want to run simple Tcl commands from other scripting languages.

For example, the following command runs the Tcl command that prints out the commands available in the project package.

```
quartus_sh --tcl_eval help -pkg project
```

## 2.4. Tcl Design Flow Controls

You can use Tcl scripts to control all aspects of the design flow, including controlling other software, when the other software also includes a scripting interface.

Typically, EDA tools include their own script interpreters that extend core language functionality with tool-specific commands. For example, the Intel Quartus Prime Tcl interpreter supports all core Tcl commands, and adds numerous commands specific to the Intel Quartus Prime software. You can include commands in one Tcl script to run another script, which allows you to combine or chain together scripts to control different tools. Because scripts for different tools must be executed with different Tcl interpreters, it is difficult to pass information between the scripts unless one script writes information into a file and another script reads it.

Within the Intel Quartus Prime software, you can perform many different operations in a design flow (such as synthesis, fitting, and timing analysis) from a single script, making it easy to maintain global state information and pass data between the operations. However, there are some limitations on the operations you can perform in a single script due to the various packages supported by each executable.

There are no limitations on running flows from any executable. Flows include operations found in

**Processing > Start** in the Intel Quartus Prime GUI, and are also documented as options for the `execute_flow` Tcl command. If you can make settings in the Intel Quartus Prime software and run a flow to get your desired result, you can make the same settings and run the same flow in a Tcl script.

### 2.4.1. Creating Projects and Making Assignments

You can create a script that makes all the assignments for an existing project, and then use the script at any time to restore your project settings to a known state.

Click **Project > Generate Tcl File for Project** to automatically generate a `.tcl` file containing your assignments. You can source this file to recreate your project, and you can add other commands to this file, such as commands for compiling the design. This file is a good starting point to learn about project management and assignment commands.

To commit the assignments you create or modify to the `.qsf` file, you use the `export_assignments` or `project_close` commands. However, when you run the `execute_flow` command, Intel Quartus Prime software automatically commits the assignment changes to the `.qsf` file. To prevent this behavior, specify the `-dont_export_assignments` logic option.

### Related Information

- Intel Quartus Prime Pro Edition Settings File Reference Manual
- Interactive Shell Mode on page 23
- Constraining Designs

## 2.4.2. Compiling Designs

You can run the Intel Quartus Prime command-line executables from Tcl scripts. Use the included `flow` package to run various Intel Quartus Prime compilation flows, or run each executable directly.

### 2.4.2.1. The `flow` Package

The `flow` package includes two commands for running Intel Quartus Prime command-line executables, either individually or together in standard compilation sequence.

- The `execute_module` command allows you to run an individual Intel Quartus Prime command-line executable.
- The `execute_flow` command allows you to run some or all the executables in commonly-used combinations.

Use the `flow` package instead of system calls to run Intel Quartus Prime executables from scripts or from the Intel Quartus Prime Tcl Console.

### 2.4.2.2. Compile All Revisions

You can use a simple Tcl script to compile all revisions in your project. Save the following script in a file called `compile_revisions.tcl` and type the following to run it:

```
quartus_sh -t compile_revisions.tcl <project name>
```

#### Compile All Revisions

```
load_package flow
project_open [lindex $quartus(args) 0]
set original_revision [get_current_revision]
foreach revision [get_project_revisions]
{
    set_current_revision $revision
    execute flow -compile
}
set_current_revision $original_revision
project_close
```

## 2.4.3. Reporting

You can extract information from the Compilation Report to evaluate results. The Intel Quartus Prime Tcl API provides easy access to report data so you do not have to write scripts to parse the text report files.

If you know the exact report cell or cells you want to access, use the `get_report_panel_data` command and specify the row and column names (or `x` and `y` coordinates) and the name of the appropriate report panel. You can often search for data in a report panel. To do this, use a loop that reads the report one row at a time with the `get_report_panel_row` command.

Column headings in report panels are in row 0. If you use a loop that reads the report one row at a time, start with row 1 to skip column headings. The `get_number_of_rows` command returns the number of rows in the report panel, including the column heading row. Since the number of rows includes the column heading row, continue your loop if the loop index is less than the number of rows.

Report panels are hierarchically arranged and each level of hierarchy is denoted by the string "||" in the panel name. For example, the name of the Fitter Settings report panel is `Fitter||Fitter Settings` because it is in the `Fitter` folder. Panels at the highest hierarchy level do not use the "||" string. For example, the Flow Settings report panel is named `Flow Settings`.

The following Tcl code prints a list of all report panel names in your project. You can run this code with any executable that includes support for the report package.

#### Print All Report Panel Names

```
load_package report
project_open myproject
load_report
set panel_names [get_report_panel_names]
foreach panel_name $panel_names {
    post_message "$panel_name"
}
```

#### 2.4.3.1. Saving Report Data in csv Format

You can create a Comma Separated Value (.csv) file from any Intel Quartus Prime report to open with a spreadsheet editor.

The following Tcl code shows a simple way to create a .csv file with data from the Fitter panel in a report.

#### Create .csv Files from Reports

```
load_package report
project_open my-project
load_report
# This is the name of the report panel to save as a CSV file
set panel_name "Fitter||Fitter Settings"
set csv_file "output.csv"
set fh [open $csv_file w]
set num_rows [get_number_of_rows -name $panel_name]
# Go through all the rows in the report file, including the
# row with headings, and write out the comma-separated data
for { set i 0 } { $i < $num_rows } { incr i } {
    set row_data [get_report_panel_row -name $panel_name \
        -row $i]
    puts $fh [join $row_data ",."]
}
close $fh
unload_report
```

You can modify the script to use command-line arguments to pass in the name of the project, report panel, and output file to use. You can run this script example with any executable that supports the report package.

#### 2.4.4. Timing Analysis

The Intel Quartus Prime Timing Analyzer includes support for industry-standard SDC commands in the `sdc` package.

The Intel Quartus Prime software includes comprehensive Tcl APIs and SDC extensions for the Timing Analyzer in the `sta`, and `sdc_ext` packages. The Intel Quartus Prime software also includes a `tdc` package that obtains information from the Timing Analyzer.

##### Related Information

[Intel Quartus Prime Pro Edition Settings File Reference Manual](#)

### 2.5. Automating Script Execution

You can configure scripts to run automatically at various points during compilation. Use this capability to automatically run scripts that perform custom reporting, make specific assignments, and perform many other tasks.

The following three global assignments control when a script is run automatically:

- `PRE_FLOW_SCRIPT_FILE` —before a flow starts
- `POST_MODULE_SCRIPT_FILE` —after a module finishes
- `POST_FLOW_SCRIPT_FILE` —after a flow finishes

A module is another term for an Intel Quartus Prime executable that performs one step in a flow. For example, two modules are Analysis and Synthesis (`quartus_syn`), and timing analysis (`quartus_sta`).

A flow is a series of modules that the Intel Quartus Prime software runs with predefined options. For example, compiling a design is a flow that typically consists of the following steps (performed by the indicated module):

1. Analysis and Synthesis (`quartus_syn`)
2. Fitter (`quartus_fit`)
3. Assembler (`quartus_asm`)
4. Timing Analyzer (`quartus_sta`)

Other flows are described in the help for the `execute_flow` Tcl command. In addition, many commands in the **Processing** menu of the Intel Quartus Prime GUI correspond to this design flow.

To make an assignment automatically run a script, add an assignment with the following form to the `.qsf` for your project:

```
set_global_assignment -name <assignment name> <executable>:<script name>
```

The Intel Quartus Prime software runs the scripts.

```
<executable> -t <script name> <flow or module name> <project name> <revision name>
```

The first argument passed in the `argv` variable (or `quartus(args)` variable) is the name of the flow or module being executed, depending on the assignment you use. The second argument is the name of the project and the third argument is the name of the revision.

The last process, current project, and current revision are passed to the script by the Intel Quartus Prime software and can be accessed by the following commands:

```
set process [lindex $quartus(args) 0]
set project [lindex $quartus(args) 1]
set revision [lindex $quartus(args) 2]

project_open $project -revision $revision
```

When you use the `POST_MODULE_SCRIPT_FILE` assignment, the specified script is automatically run after every executable in a flow. You can use a string comparison with the module name (the first argument passed in to the script) to isolate script processing to certain modules.

### 2.5.1. Execution Example

To illustrate how automatic script execution works in a complete flow, assume you have a project called `top` with a current revision called `rev_1`, and you have the following assignments in the `.qsf` for your project.

```
set_global_assignment -name PRE_FLOW_SCRIPT_FILE quartus_sh:first.tcl
set_global_assignment -name POST_MODULE_SCRIPT_FILE quartus_sh:next.tcl
set_global_assignment -name POST_FLOW_SCRIPT_FILE quartus_sh:last.tcl
```

When you compile your project, the `PRE_FLOW_SCRIPT_FILE` assignment causes the following command to be run before compilation begins:

```
quartus_sh -t first.tcl compile top rev_1
```

Next, the Intel Quartus Prime software starts compilation with analysis and synthesis, performed by the `quartus_syn` executable. After the Analysis and Synthesis finishes, the `POST_MODULE_SCRIPT_FILE` assignment causes the following command to run:

```
quartus_sh -t next.tcl quartus_syn top rev_1
```

Then, the Intel Quartus Prime software continues compilation with the Fitter, performed by the `quartus_fit` executable. After the Fitter finishes, the `POST_MODULE_SCRIPT_FILE` assignment runs the following command:

```
quartus_sh -t next.tcl quartus_fit top rev_1
```

Corresponding commands are run after the other stages of the compilation. When the compilation is over, the `POST_FLOW_SCRIPT_FILE` assignment runs the following command:

```
quartus_sh -t last.tcl compile top rev_1
```

## 2.5.2. Controlling Processing

The POST\_MODULE\_SCRIPT\_FILE assignment causes a script to run after every module. Because the same script is run after every module, you might have to include some conditional statements that restrict processing in your script to certain modules.

For example, if you want a script to run only after timing analysis, use a conditional test like the following example. It checks the flow or module name passed as the first argument to the script and executes code when the module is quartus\_sta.

### Restrict Processing to a Single Module

```
set module [lindex $quartus(args) 0]
if [string match "quartus_sta" $module] {
    # Include commands here that are run
    # after timing analysis
    # Use the post-message command to display
    # messages
    post_message "Running after timing analysis"
}
```

## 2.5.3. Displaying Messages

Because of the way the Intel Quartus Prime software runs the scripts automatically, you must use the post\_message command to display messages, instead of the puts command. This requirement applies only to scripts that are run by the three assignments listed in "Automating Script Execution".

### Related Information

- [The post\\_message Command](#) on page 40
- [Automating Script Execution](#) on page 27

## 2.6. Other Scripting Features

The Intel Quartus Prime Tcl API includes other general-purpose commands and features described in this section.

### 2.6.1. Natural Bus Naming

The Intel Quartus Prime software supports natural bus naming. Natural bus naming allows you to use square brackets to specify bus indexes in HDL, without including escape characters to prevent Tcl from interpreting the square brackets as containing commands. For example, one signal in a bus named address can be identified as address[0] instead of address\[0\]. You can take advantage of natural bus naming when making assignments.

```
set_location_assignment -to address[10] Pin_M20
```

The Intel Quartus Prime software defaults to natural bus naming. You can turn off natural bus naming with the disable\_natural\_bus\_naming command. For more information about natural bus naming, type the following at an Intel Quartus Prime Tcl prompt:

```
enable_natural_bus_naming -h
```

## 2.6.2. Short Option Names

You can use short versions of command options, if they are unambiguous. For example, the `project_open` command supports two options: `-current_revision` and `-revision`.

You can use any of the following abbreviations of the `-revision` option:

- `-r`
- `-re`
- `-rev`
- `-revi`
- `-revis`
- `-revisio`

You can use an extremely short option such as `-r` because in the case of the `project_open` command no other option starts with the letter r. However, the `report_timing` command includes the options `-recovery` and `-removal`. You cannot use `-r` or `-re` to shorten either of those options, because the abbreviation is not unique.

## 2.6.3. Collection Commands

Some Intel Quartus Prime Tcl functions return very large sets of data that are inefficient as Tcl lists. These data structures are referred to as collections. The Intel Quartus Prime Tcl API uses a collection ID to access the collection.

There are two Intel Quartus Prime Tcl commands for working with collections, `foreach_in_collection` and `get_collection_size`. Use the `set` command to assign a collection ID to a variable.

### 2.6.3.1. The `foreach_in_collection` Command

The `foreach_in_collection` command is similar to the `foreach` Tcl command. Use it to iterate through all elements in a collection. The following example prints all instance assignments in an open project.

#### `foreach_in_collection` Example

```
set all_instance_assignments [get_all_instance_assignments -name *]
foreach_in_collection asgn $all_instance_assignments {
    # Information about each assignment is
    # returned in a list. For information
    # about the list elements, refer to Help
    # for the get-all-instance-assignments command.
    set to [lindex $asgn 2]
    set name [lindex $asgn 3]
    set value [lindex $asgn 4]
    puts "Assignment to $to: $name = $value"
}
```

#### Related Information

[foreach\\_in\\_collection \(::quartus::misc\)](#)  
In Intel Quartus Prime Help

### 2.6.3.2. The `get_collection_size` Command

Use the `get_collection_size` command to get the number of elements in a collection. The following example prints the number of global assignments in an open project.

#### `get_collection_size` Example

```
set all_global_assignments [get_all_global_assignments -name *]
set num_global_assignments [get_collection_size $all_global_assignments]
puts "There are $num_global_assignments global assignments in your project"
```

## 2.6.4. Node Finder Commands

The Node Finder allows you to find any node name in your project's compilation database. You can then perform various actions on found nodes, such as specifying constraints or assignments to those nodes. You can filter the search on various criteria, and also use wildcard characters in the search string.

A complete set of Node Finder Tcl commands that support the equivalent Node Finder filtering options is available for use in the scripted design flow environment.

The filtering options include the following default filters that appear in the filter combo box in the Node Finder:

- [Design Entry \(all names\) Filter](#) on page 31
- [Pins: assigned Filter](#) on page 32
- [Pins: unassigned Filter](#) on page 32
- [Pins: input Filter](#) on page 32
- [Pins: output Filter](#) on page 33
- [Pins: bidirectional Filter](#) on page 33
- [Pins: virtual Filter](#) on page 33
- [Pins: all Filter](#) on page 34
- [Pins: all & Registers: post-fitting Filter](#) on page 34
- [Ports: partition](#) on page 35
- [Entity instance: pre-synthesis Filter](#) on page 35
- [Registers: pre-synthesis Filter](#) on page 35
- [Registers: post-fitting Filter](#) on page 36
- [Post-synthesis Filter](#) on page 36
- [Post-Compilation Filter](#) on page 36
- [Signal Tap: pre-synthesis Filter](#) on page 37
- [Signal Tap: post-fitting Filter](#) on page 37

### 2.6.4.1. Design Entry (all names) Filter

This Node Finder filter finds all user-entered names in your design.

The following Tcl command demonstrates the use of the Design Entry (all names) filtering option:

```
set name_ids_col [get_names -filter * -node_type all \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.2. Pins: assigned Filter

This Node Finder filter finds all pin names assigned locations or other pin-related assignments.

The following Tcl command demonstrates the use of the Pins: assigned filtering option:

```
set name_ids_col [get_names -filter * -node_type assigned \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.3. Pins: unassigned Filter

This Node Finder filter finds all pin names unassigned locations or other pin related assignments.

The following Tcl command demonstrates the use of the Pins: unassigned filtering option:

```
set name_ids_col [get_names -filter * -node_type unassigned \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.4. Pins: input Filter

This Node Finder filter finds all input pin names in your design files.

The following Tcl command demonstrates the use of the Pins: input filtering option:

```
set name_ids_col [get_names -filter * -node_type input \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.5. Pins: output Filter

This Node Finder filter finds all output pin names in your design files.

The following Tcl command demonstrates the use of the Pins: output filtering option:

```
set name_ids_col [get_names -filter * -node_type output \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.6. Pins: bidirectional Filter

This Node Finder filter finds all bidirectional pin names in your design files.

The following Tcl command demonstrates the use of the Pins: bidirectional filtering option:

```
set name_ids_col [get_names -filter * -node_type bidir \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.7. Pins: virtual Filter

This Node Finder filter finds names of all I/O elements mapped to logic elements with a Virtual Pin logic option assignment.

The following Tcl command demonstrates the use of the Pins: virtual filtering option:

```
set name_ids_col [get_names -filter * -node_type virtual \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.8. Pins: all Filter

This Node Finder filter finds all pin names in your design files.

The following Tcl command demonstrates the use of the Pins: all filtering option:

```
set name_ids_col [get_names -filter * -node_type \
pin -observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.9. Pins: all & Registers: post-fitting Filter

This Node Finder filter finds all pin names in your design along with all register names from your design files that persist after physical synthesis and fitting. The Pins: all & Registers: post-fitting filter is a combination of the Pins: all and Registers: post-fitting filters.

The following Tcl command demonstrates the use of the Pins: all & Registers: post-fitting filtering option:

```
set name_ids_col [get_names -filter * -node_type \
all_reg -observable_type post_fitter]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type post_fitter \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.10. Ports: partition

This Node Finder filter must be used after running the Fitter, to find nodes for post-fit partition.

**Note:** When you run this filter before running the Fitter, a "No nodes available. Run Fitter." message displays.

The following Tcl command demonstrates the use of the Ports: partition filtering option:

```
set name_ids_col [get_names -filter * -node_type partition \
-observable_type post_fitter]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type post_fitter \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the get\_names command, refer to [The get\\_names Command](#) on page 38.

#### 2.6.4.11. Entity instance: pre-synthesis Filter

This Node Finder filter finds a list of instances in the logical hierarchy for pre-synthesis netlist.

The following Tcl command demonstrates the use of the Entity instance: pre-synthesis filtering option:

```
set name_ids_col [get_names -filter * -node_type hierarchy \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the get\_names command, refer to [The get\\_names Command](#) on page 38.

#### 2.6.4.12. Registers: pre-synthesis Filter

This Node Finder filter finds all register names you entered in the design after Analysis and Elaboration, but before physical synthesis performs any synthesis optimizations.

The following Tcl command demonstrates the use of the Registers: pre-synthesis filtering option:

```
set name_ids_col [get_names -filter * -node_type reg \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type pre_synthesis \
$name_id]
    append name ","
```

```
        append name [get_name_info -info node_type $name_id]
        puts $name
    }
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.13. Registers: post-fitting Filter

This Node Finder filter finds all user-entered register names in your design files that remain after physical synthesis and fitting.

The following Tcl command demonstrates the use of the Registers: post-fitting filtering option:

```
set name_ids_col [get_names -filter * -node_type reg \
-observable_type post_fitter]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type post_fitter \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.14. Post-synthesis Filter

This Node Finder filter finds all user-entered and synthesis-generated names that remain in the design after design elaboration and physical synthesis.

The following Tcl command demonstrates the use of the Post-Synthesis filtering option:

```
set name_ids_col [get_names -filter * -node_type all \
-observable_type post_synthesis]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type post_synthesis \
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.15. Post-Compilation Filter

This Node Finder filter finds all user-centered and Compiler-generated names that remain after fitting and do not have location assignments.

The following Tcl command demonstrates the use of the Post-Compilation filtering option:

```
set name_ids_col [get_names -filter * -node_type all \
-observable_type post_fitter]
foreach_in_collection name_id $name_ids_col {
    set name [get_name_info -info full_path -observable_type post_fitter \

```

```
$name_id]
    append name ","
    append name [get_name_info -info node_type $name_id]
    puts $name
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.16. Signal Tap: pre-synthesis Filter

This Node Finder filter finds all internal device nodes in the pre-synthesis netlist that can be analyzed by the Signal Tap Logic Analyzer.

The following Tcl command demonstrates the use of the `Signal Tap: pre-synthesis` filtering option:

```
set name_ids_col [get_names -filter * -node_type all \
-observable_type pre_synthesis]
foreach_in_collection name_id $name_ids_col {
    set is_signaltap [get_name_info -info signaltapii -observable_type \
pre_synthesis $name_id]
    if {$is_signaltap == 1} {
        set name [get_name_info -use_cached_database -info full_path \
-observable_type pre_synthesis $name_id]
        append name ","
        append name [get_name_info -info node_type -observable_type \
pre_synthesis $name_id]
        puts $name
    }
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

#### 2.6.4.17. Signal Tap: post-fitting Filter

This Node Finder filter finds all internal device nodes in the post fit netlist that can be analyzed by the Signal Tap Logic Analyzer.

The following Tcl command demonstrates the use of the `Signal Tap: post-fitting` filtering option:

```
set name_ids_col [get_names -filter * -node_type all \
-observable_type post_fitter]
foreach_in_collection name_id $name_ids_col {
    set is_signaltap [get_name_info -info signaltapii -observable_type \
post_fitter $name_id]
    if {$is_signaltap == 1} {
        set name [get_name_info -use_cached_database -info full_path \
-observable_type post_fitter $name_id]
        append name ","
        append name [get_name_info -info node_type -observable_type \
pre_synthesis $name_id]
        puts $name
    }
}
```

For more information about the `get_names` command, refer to [The `get\_names` Command](#) on page 38.

## 2.6.5. The get\_names Command

To query a filtered output collection of all matching node name IDs found in a compiled Intel Quartus Prime project, use the `get_names` command.

To access each element of the output collection, use the Tcl command `foreach_in_collection`. For `get_names` or `foreach_in_collection` command example, type `get_names -long_help` or `foreach_in_collection -long_help`.

- If the `-node_type` option is not specified, the default value is `all`.
- If the `-observable_type` option is not specified, the default value is `all`.
- The node type `pin` includes `input`, `output`, `bidir`, `assigned`, `unassigned`, `virtual`, and `pin`.
- The node type `qsf` include names from the `.qsf` settings file.
- The node type `all` includes all node types.
- The node type `all_reg` includes all node types and registers post-fitting.

The value for `-observable_type` option can be one of the following:

**Table 9. Values for observable\_type Option**

Observable Type	Description
<code>all</code>	Use post-Fitter information. If it is not available, post-synthesis information is used. Else, pre-synthesis information is used if it exists.
<code>pre_synthesis</code>	Use pre-synthesis information.
<code>post_synthesis</code>	Use post-synthesis information.
<code>post_fitter</code>	Use post-Fitter information.
<code>post_asm</code>	Use post-Assembler information. The post-Assembler information is supported only for designs using the HardCopy II device family.
<code>stp_pre_synthesis</code>	Use Signal Tap pre-synthesis information.

### Arguments

Following table lists the `get_names` command arguments:

**Table 10. The get\_names Command Arguments**

Argument	Description
<code>-h   -help</code>	Displays a short help.
<code>-long help</code>	Displays a long help with examples and possible return values.
<code>-entity&lt;wildcard&gt;</code>	Specifies the entity to get names from hierarchies instantiated by the entity.

*continued...*

Argument	Description
<code>-filter&lt;wildcard&gt;</code>	Specifies the node's full path name and wildcard characters.
<code>-node_type &lt;all comb reg pin input output bidir hierarchy mem bus qsf state_machine assigned unassigned all_reg partition virtual&gt;</code>	Filters based on the specified node type.
<code>-observable_type &lt;all pre_synthesis post_synthesis post_fitter stp_pre_synthesis&gt;]</code>	Filters based on the specified observable type.

### Return Values

Following table lists values returned by the `get_names` command:

**Table 11. The `get_names` Command Return Values**

Code Name	Code	String Returned
TCL_OK	0	INFO: operation successful
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Get names cannot return <string> because the name was found in a partition that's not the root partition. Refine your <code>get_names</code> search pattern to exclude child partitions.
TCL_ERROR	1	ERROR: Compiler database does not exist for revision name: <string>. At the minimum, run Analysis & Synthesis with the specified revision name before using this Tcl command.
TCL_ERROR	1	ERROR: Illegal node type: <string>. Specify "all", "comb", "reg", "pin", "hierarchy", or "bus".
TCL_ERROR	1	ERROR: Illegal observable type: <string>. Specify "all", "pre_synthesis", "post_synthesis", or "post_fitter".
TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### Example Use

```
# Search for a single post-Fitter pin with the name accel and make assignments
set accel_name_id [get_names -filter accel -node_type pin -observable_type
post_fitter]

foreach_in_collection name_id $accel_name_id {
    # Get the full path name of the node
    set target [get_name_info -info full_path $name_id]
    # Set multicycle assignment
    set_multicycle_assignment -to $target 2
    # Set location assignment
    set_location_assignment -to $target Pin_E22
}
# Search for nodes of any post-Fitter node type with name length <= 5. The
# default node type is "all"
set name_ids [get_names -filter ???? -observable_type post_fitter]
foreach_in_collection name_id $name_ids {
    # Print the name id
    puts $name_id
    # Print the node type
    puts [get_name_info -info node_type $name_id]
    # Print the full path (which excludes the current focus entity from the path)
    puts [get_name_info -info full_path $name_id]
}
# Search for nodes of any post-Fitter node type that end in "eed".
```

```
# The default node type is "all"
set name_ids [get_names -filter *eed -observable_type post_fitter]
foreach_in_collection name_id $name_ids {
    # Print the name id
    puts $name_id
    # Print the node type
    puts [get_name_info -info node_type $name_id]
    # Print the full path (which excludes the current
    # focus entity from the path)
    puts [get_name_info -info full_path $name_id]
}
```

## 2.6.6. The `post_message` Command

To print messages that are formatted like Intel Quartus Prime software messages, use the `post_message` command. Messages printed by the `post_message` command appear in the **System** tab of the **Messages** window in the Intel Quartus Prime GUI, and are written to standard output when scripts are run. Arguments for the `post_message` command include an optional message type and a required message string.

The message type can be one of the following:

- `info` (default)
- `extra_info`
- `warning`
- `critical_warning`
- `error`

If you do not specify a type, Intel Quartus Prime software defaults to `info`.

With the Intel Quartus Prime software in Windows, you can color code messages displayed at the system command prompt with the `post_message` command. Add the following line to your `quartus2.ini` file:

```
DISPLAY_COMMAND_LINE_MESSAGES_IN_COLOR = on
```

The following example shows how to use the `post_message` command.

```
post_message -type warning "Design has gated clocks"
```

## 2.6.7. Accessing Command-Line Arguments

The global variable `quartus(args)` is a list of the arguments typed on the command-line following the name of the Tcl script.

### Example 1. Simple Command-Line Argument Access

The following Tcl example prints all the arguments in the `quartus(args)` variable:

```
set i 0
foreach arg $quartus(args) {
    puts "The value at index $i is $arg"
    incr i
}
```

## Example 2. Passing Command-Line Arguments to Scripts

If you copy the script in the previous example to a file named `print_args.tcl`, it displays the following output when you type the following at a command prompt.

```
quartus_sh -t print_args.tcl my_project 100MHz
The value at index 0 is my_project
The value at index 1 is 100MHz
```

### 2.6.7.1. The cmdline Package

You can use the `cmdline` package included with the Intel Quartus Prime software for more robust and self-documenting command-line argument passing. The `cmdline` package supports command-line arguments with the form `-<option><value>`.

#### cmdline Package

```
package require cmdline
variable ::argv0 $::quartus(args)
set options {
    { "project.arg" "" "Project name" }
    { "frequency.arg" "" "Frequency" }
}
set usage "You need to specify options and values"
array set optshash [::cmdline::getoptions ::argv $options $usage]
puts "The project name is $optshash(project)"
puts "The frequency is $optshash(frequency)"
```

If you save those commands in a Tcl script called `print_cmd_args.tcl` you see the following output when you type the following command at a command prompt.

#### Passing Command-Line Arguments for Scripts

```
quartus_sh -t print_cmd_args.tcl -project my_project -frequency 100MHz
The project name is my_project
The frequency is 100MHz
```

Virtually all Intel Quartus Prime Tcl scripts must open a project. You can open a project, and you can optionally specify a revision name with code like the following example. The example checks whether the specified project exists. If it does, the example opens the current revision, or the revision you specify.

#### Full-Featured Method to Open Projects

```
package require cmdline
variable ::argv0 $::quartus(args)
set options {
    { "project.arg" "" "Project Name" } \
    { "revision.arg" "" "Revision Name" } \
}
array set optshash [::cmdline::getoptions ::argv0 $options]
# Ensure the project exists before trying to open it
if {[project_exists $optshash(project)]} {
    if {[string equal "" $optshash(revision)]} {
        # There is no revision name specified, so default
        # to the current revision
        project_open $optshash(project) -current_revision
    } else {
        # There is a revision name specified, so open the
        # project with that revision
        project_open $optshash(project) -revision \
            $optshash(revision)
    }
}
```

```
    } else {
        puts "Project $optshash(project) does not exist"
        exit 1
    }
# The rest of your script goes here
```

If you do not require this flexibility or error checking, you can use just the `project_open` command.

### Simple Method to Open Projects

```
set proj_name [lindex $argv 0]
project_open $proj_name
```

## 2.6.8. The `quartus()` Array

The global `quartus()` Tcl array includes other information about your project and the current Intel Quartus Prime executable that might be useful to your scripts. The scripts in the preceding examples parsed command line arguments found in `quartus(args)`. For information on the other elements of the `quartus()` array, type the following command at a Tcl prompt:

```
help -quartus
```

## 2.7. The Intel Quartus Prime Tcl Shell in Interactive Mode Example

This section presents how to make project assignments and then compile the finite impulse response (FIR) filter tutorial project with the `quartus_sh` interactive shell.

This example assumes you already have the `fir_filter` tutorial design files in a project directory.

1. To run the interactive Tcl shell, type the following at the system command prompt:

```
quartus_sh -s
```

2. Create a new project called `fir_filter`, with a revision called `filtref` by typing:

```
project_new -revision filtref fir_filter
```

*Note:* • If the project file and project name are the same, the Intel Quartus Prime software gives the revision the same name as the project.  
• If a `.qpf` file for this project already exists, the Intel Quartus Prime software will display an error stating that the project already exists.

Because the revision named `filtref` matches the top-level file, all design files are automatically picked up from the hierarchy tree.

3. Set a global assignment for the device:

```
set_global_assignment -name family <device family name>
```

To learn more about assignment names that you can use with the `-name` option, refer to Intel Quartus Prime Help.

*Note:* For assignment values that contain spaces, enclose the value in quotation marks.

4. To compile a design, use the `::quartus::flow` package, which properly exports the new project assignments and compiles the design with the proper sequence of the command-line executables. First, load the package:

```
load_package flow
```

It returns:

```
1.1
```

5. To perform a full compilation of the FIR filter design, use the `execute_flow` command with the `-compile` option:

```
execute_flow -compile
```

This command compiles the FIR filter tutorial project, exporting the project assignments and running `quartus_syn`, `quartus_fit`, `quartus_asm`, and `quartus_sta`. This sequence of events is the same as selecting **Processing > Start Compilation** in the Intel Quartus Prime GUI.

6. When you are finished with a project, close it with the `project_close` command.
7. To exit the interactive Tcl shell, type `exit` at a Tcl prompt.

## 2.8. The tclsh Shell

On the UNIX and Linux operating systems, the tclsh shell included with the Intel Quartus Prime software is initialized with a minimal PATH environment variable. As a result, system commands might not be available within the tclsh shell because certain directories are not in the PATH environment variable.

To include other directories in the path searched by the tclsh shell, set the `QUARTUS_INIT_PATH` environment variable before running the tclsh shell. Directories in the `QUARTUS_INIT_PATH` environment variable are searched by the tclsh shell when you execute a system command.

## 2.9. Tcl Scripting Basic Examples

The core Tcl commands support variables, control structures, and procedures. Additionally, there are commands for accessing the file system and network sockets, and running other programs. You can create platform-independent graphical interfaces with the Tk widget set.

Tcl commands are executed immediately as they are typed in an interactive Tcl shell. You can also create scripts (including the examples in this chapter) in files and run them with the Intel Quartus Prime executables or with the tclsh shell.

### 2.9.1. Hello World Example

The following shows the basic “Hello world” example in Tcl:

```
puts "Hello world"
```

Use double quotation marks to group the words `hello` and `world` as one argument. Double quotation marks allow substitutions to occur in the group. Substitutions can be simple variable substitutions, or the result of running a nested command. Use curly braces `{ }` for grouping when you want to prevent substitutions.

## 2.9.2. Variables

Assign a value to a variable with the `set` command. You do not have to declare a variable before using it. Tcl variable names are case-sensitive.

```
set a 1
```

To access the contents of a variable, use a dollar sign ("`$`") before the variable name. The following example prints "Hello world" in a different way.

```
set a Hello
set b world
puts "$a $b"
```

## 2.9.3. Substitutions

Tcl performs three types of substitution:

- Variable value substitution
- Nested command substitution
- Backslash substitution

### 2.9.3.1. Variable Value Substitution

Variable value substitution, refers to accessing the value stored in a variable with a dollar sign ("`$`") before the variable name.

### 2.9.3.2. Nested Command Substitution

Nested command substitution refers to how the Tcl interpreter evaluates Tcl code in square brackets. The Tcl interpreter evaluates nested commands, starting with the innermost nested command, and commands nested at the same level from left to right. Each nested command result is substituted in the outer command.

```
set a [string length foo]
```

### 2.9.3.3. Backslash Substitution

Backslash substitution allows you to quote reserved characters in Tcl, such as dollar signs ("`$`") and braces ("[ ]"). You can also specify other special ASCII characters like tabs and new lines with backslash substitutions. A backslash before a character tells the TCL interpreter to treat the next character as a literal if the character is not the last character on the line.

```
puts "This is a \$ special character"
puts "This is a \
$ special character and line continuation"
puts "This is backslash \is ignored"
```

```
puts "This is backslash\
continued on next line"
```

## 2.9.4. Arithmetic

Use the `expr` command to perform arithmetic calculations. Use curly braces ("{}") to group the arguments of this command for greater efficiency and numeric precision.

```
set a 5
set b [expr { $a + sqrt(2) }]
```

The Intel Quartus Prime software supports all standard Tcl boolean and arithmetic operators, such as `&&` (AND), `||` (OR), `!` (NOT), and comparison operators such as `<` (less than), `>` (greater than), and `==` (equal to).

## 2.9.5. Lists

A Tcl list is a series of values. Supported list operations include creating lists, appending lists, extracting list elements, computing the length of a list, sorting a list, and more.

```
set a { 1 2 3 }
```

You can use the `lindex` command to extract information at a specific index in a list. Indexes are zero-based. You can use the `index end` to specify the last element in the list, or the `index end-<n>` to count from the end of the list. For example, to print the second element (at index 1) in the list stored in `a` use the following code.

```
puts [lindex $a 1]
```

The `llength` command returns the length of a list.

```
puts [llength $a]
```

The `lappend` command appends elements to a list. If a list does not already exist, the list you specify is created. The list variable name is not specified with a dollar sign ("\$").

```
lappend a 4 5 6
```

## 2.9.6. Arrays

Arrays are similar to lists except that they use a string-based index. Tcl arrays are implemented as hash tables. You can create arrays by setting each element individually or with the `array set` command.

To set an element with an index of `Mon` to a value of `Monday` in an array called `days`, use the following command:

```
set days(Mon) Monday
```

The `array set` command requires a list of index/value pairs. This example sets the array called `days`:

```
array set days { Sun Sunday Mon Monday Tue Tuesday \
    Wed Wednesday Thu Thursday Fri Friday Sat Saturday }
set day_abbreviation Mon
puts $days($day_abbreviation)
```

Use the `array names` command to get a list of all the indexes in a particular array. The index values are not returned in any specified order. The following example is one way to iterate over all the values in an array.

```
foreach day [array names days] {
    puts "The abbreviation $day corresponds to the day name $days($day)"
}
```

Arrays are a very flexible way of storing information in a Tcl script and are a good way to build complex data structures.

## 2.9.7. Control Structures

Tcl supports common control structures, including if-then-else conditions and for, foreach, and while loops. The position of the curly braces as shown in the following examples ensures the control structure commands are executed efficiently and correctly. The following example prints whether the value of variable `a` positive, negative, or zero.

### If-Then-Else Structure

```
if { $a > 0 } { puts "The value is positive"
} elseif { $a < 0 } {
    puts "The value is negative"
} else {
    puts "The value is zero"
}
```

The following example uses a for loop to print each element in a list.

### For Loop

```
set a { 1 2 3 }
for { set i 0 } { $i < [llength $a] } { incr i }
    puts "The list element at index $i is [lindex $a $i]"
}
```

The following example uses a foreach loop to print each element in a list.

### foreach Loop

```
set a { 1 2 3 }
foreach element $a {
    puts "The list element is $element"
}
```

The following example uses a while loop to print each element in a list.

### while Loop

```
set a { 1 2 3 }
set i 0
while { $i < [llength $a] } { puts "The list element at index $i is [lindex $a
$i]"
    incr i
}
```

You do not have to use the `expr` command in boolean expressions in control structure commands because they invoke the `expr` command automatically.

## 2.9.8. Procedures

Use the `proc` command to define a Tcl procedure (known as a subroutine or function in other scripting and programming languages). The scope of variables in a procedure is local to the procedure. If the procedure returns a value, use the `return` command to return the value from the procedure. The following example defines a procedure that multiplies two numbers and returns the result.

### Simple Procedure

```
proc multiply { x y } {
    set product [expr { $x * $y }]
    return $product
}
```

The following example shows how to use the `multiply` procedure in your code. You must define a procedure before your script calls it.

### Using a Procedure

```
proc multiply { x y } {
    set product [expr { $x * $y }]
    return $product
}
set a 1
set b 2
puts [multiply $a $b]
```

Define procedures near the beginning of a script. If you want to access global variables in a procedure, use the `global` command in each procedure that uses a global variable.

### Accessing Global Variables

```
proc print_global_list_element { i } {
    global my_data
    puts "The list element at index $i is [lindex $my_data $i]"
}
set my_data { 1 2 3 }
print_global_list_element 0
```

## 2.9.9. File I/O

Tcl includes commands to read from and write to files. You must open a file before you can read from or write to it, and close it when the read and write operations are done.

To open a file, use the `open` command; to close a file, use the `close` command. When you open a file, specify its name and the mode in which to open it. If you do not specify a mode, Tcl defaults to read mode. To write to a file, specify `w` for write mode.

### Open a File for Writing

```
set output [open myfile.txt w]
```

Tcl supports other modes, including appending to existing files and reading from and writing to the same file.

The `open` command returns a file handle to use for read or write access. You can use the `puts` command to write to a file by specifying a file handle.

### Write to a File

```
set output [open myfile.txt w]
puts $output "This text is written to the file."
close $output
```

You can read a file one line at a time with the `gets` command. The following example uses the `gets` command to read each line of the file and then prints it out with its line number.

### Read from a File

```
set input [open myfile.txt]
set line_num 1
while { [gets $input line] >= 0 } {
    # Process the line of text here
    puts "$line_num: $line"
    incr line_num
}
close $input
```

## 2.9.10. Syntax and Comments

Arguments to Tcl commands are separated by white space, and Tcl commands are terminated by a newline character or a semicolon. You must use backslashes when a Tcl command extends more than one line. The backslash (\) must be the last character in the line to designate line extension. If the backslash is followed by any other character including a space, that character is treated as a literal.

Tcl uses the hash or pound character (#) to begin comments. The # character must begin a comment. If you prefer to include comments on the same line as a command, be sure to terminate the command with a semicolon before the # character. The following example is a valid line of code that includes a `set` command and a comment.

```
set a 1;# Initializes a
```

Without the semicolon, the command is invalid because the `set` command does not terminate until the new line after the comment.

The Tcl interpreter counts curly braces inside comments, which can lead to errors that are difficult to track down. The following example causes an error because of unbalanced curly braces.

```
# if { $x > 0 } {  
if { $y > 0 } {  
    # code here  
}
```

## 2.9.11. External References

For more information about Tcl, refer to the following sources:

- Brent B. Welch and Ken Jones, and Jeffery Hobbs, *Practical Programming in Tcl and Tk* (Upper Saddle River: Prentice Hall, 2003)
- John Ousterhout and Ken Jones, *Tcl and the Tk Toolkit* (Boston: Addison-Wesley Professional, 2009)
- Mark Harrison and Michael McLennan, *Effective Tcl/Tk Programming: Writing Better Programs in Tcl and Tk* (Boston: Addison-Wesley Professional, 1997)

### Related Information

[www.tcl.tk](http://www.tcl.tk)  
Tcl Developer Xchange

## 2.10. Tcl Scripting Revision History

The following revision history applies to this chapter:

**Table 12. Document Revision History**

Document Version	Intel Quartus Prime Version	Changes
2020.12.14	20.4	<ul style="list-style-type: none"><li>Revised "Tcl Scripting" topic to include link to new "Tcl Commands and Packages" reference.</li><li>Revised "Tcl Packages" topic for latest supported packages.</li></ul>
2019.06.28	19.1	Minor correction in <i>The get_names Command</i>
2019.04.01	19.1	<ul style="list-style-type: none"><li>Added Node Finder Tcl commands.</li><li>Added the <code>get_names</code> command.</li><li>Rectified code snippet formatting in <i>Compile All Revisions, Arrays, and Control Structures</i> topics.</li></ul>
2018.05.07	18.0.0	<ul style="list-style-type: none"><li>Removed deprecated options.</li><li>External reference links updated.</li><li>Corrected typos and made minor content fixes.</li></ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"><li>Implemented Intel rebranding.</li></ul>
2015.11.02	15.1.0	<ul style="list-style-type: none"><li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li><li>Updated the list of Tcl packages in the <i>Quartus Prime Tcl Packages</i> section.</li><li>Updated the <i>Quartus Prime Tcl API Help</i> section:<ul style="list-style-type: none"><li>Updated the Tcl Help Output</li></ul></li></ul>
June 2014	14.0.0	Updated the format.
June 2012	12.0.0	<ul style="list-style-type: none"><li>Removed survey link.</li></ul>

*continued...*

Document Version	Intel Quartus Prime Version	Changes
November 2011	11.0.1	<ul style="list-style-type: none"><li>• Template update</li><li>• Updated supported version of Tcl in the section “Tool Command Language.”</li><li>• minor editorial changes</li></ul>
May 2011	11.0.0	Minor updates throughout document.
December 2010	10.1.0	Template update Updated to remove tcl packages used by the Classic Timing Analyzer
July 2010	10.0.0	Minor updates throughout document.
November 2009	9.1.0	<ul style="list-style-type: none"><li>• Removed LogicLock example.</li><li>• Added the incremental_compilation, insystem_source_probe, and rtl packages to Table 3-1 and Table 3-2.</li><li>• Added quartus_map to table 3-2.</li></ul>
March 2009	9.0.0	<ul style="list-style-type: none"><li>• Removed the “EDA Tool Assignments” section</li><li>• Added the section “Compile All Revisions” on page 3-9</li><li>• Added the section “Using the tclsh Shell” on page 3-20</li></ul>
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated references.

## 3. TCL Commands and Packages

---

### 3.1. TCL Commands and Packages Summary

Tcl Command	Tcl Package	Package Version
get_back_annotation_assignments	backannotate	1.1
logiclock_back_annotate	backannotate	1.1
apply_assignments	bpps	1.0
check_plan	bpps	1.0
export_constraints_to_qsf	bpps	1.0
get_cell_info	bpps	1.0
get_device	bpps	1.0
get_hdbpath_from_id	bpps	1.0
get_id_from_hdbpath	bpps	1.0
get_location_info	bpps	1.0
get_placement	bpps	1.0
get_placement_info	bpps	1.0
get_placements	bpps	1.0
get_placements_of_group	bpps	1.0
harden_cell	bpps	1.0
harden_cells	bpps	1.0
initialize	bpps	1.0
load_floorplan	bpps	1.0
place_cells	bpps	1.0
read_tpl_placement	bpps	1.0
remove_invalid_reports	bpps	1.0
report_all	bpps	1.0
report_cell_connectivity	bpps	1.0
report_cell_placement_reasons	bpps	1.0
report_cells	bpps	1.0
report_clocks	bpps	1.0

*continued...*

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

\*Other names and brands may be claimed as the property of others.

ISO  
9001:2015  
Registered

Tcl Command	Tcl Package	Package Version
report_legal_cell_locations	bpps	1.0
report_location_types	bpps	1.0
report_locations	bpps	1.0
report_regions	bpps	1.0
report_summary	bpps	1.0
reset_plan	bpps	1.0
save_floorplan	bpps	1.0
save_pin_assignments	bpps	1.0
set_mode	bpps	1.0
shutdown	bpps	1.0
soften_cell	bpps	1.0
soften_cells	bpps	1.0
undo_last_placement	bpps	1.0
unplace_cells	bpps	1.0
update_pdpw	bpps	1.0
validate_placement	bpps	1.0
write_plan	bpps	1.0
write_tpl_placement	bpps	1.0
check_node	chip_planner	2.0
close_chip_planner	chip_planner	2.0
design_has_ace_support	chip_planner	2.0
design_has_encrypted_ip	chip_planner	2.0
get_info_parameters	chip_planner	2.0
get_iports	chip_planner	2.0
get_node_by_name	chip_planner	2.0
get_oports	chip_planner	2.0
get_port_by_type	chip_planner	2.0
get_sp_pin_list	chip_planner	2.0
get_tile_power_setting	chip_planner	2.0
read_netlist	chip_planner	2.0
set_batch_mode	chip_planner	2.0
commit_design	design	1.0
convert_partition	design	1.0
create_assignment	design	1.0
delete_assignments	design	1.0
disable_assignments	design	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
enable_assignments	design	1.0
export_design	design	1.0
export_partition	design	1.0
extract_metadata	design	1.0
get_assignment_info	design	1.0
get_assignment_names	design	1.0
get_assignments	design	1.0
get_entity_names	design	1.0
get_instances	design	1.0
import_design	design	1.0
import_partition	design	1.0
list_valid_snapshot_names	design	1.0
load_design	design	1.0
report_assignments	design	1.0
set_assignment_info	design	1.0
get_family_list	device	1.0
get_part_info	device	1.0
get_part_list	device	1.0
report_device_info	device	1.0
report_family_info	device	1.0
report_part_info	device	1.0
add_check_op	drc	1.0
add_check_parameter	drc	1.0
add_object	drc	1.0
add_object_with_properties	drc	1.0
add_property	drc	1.0
add_rule	drc	1.0
add_ruleViolation	drc	1.0
add_violation_record	drc	1.0
check_design	drc	1.0
delete_waivers	drc	1.0
get_objects	drc	1.0
get_option	drc	1.0
get_property	drc	1.0
get_stage_info	drc	1.0
get_waivers	drc	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
list_properties	drc	1.0
report_waivers	drc	1.0
set_option	drc	1.0
set_property	drc	1.0
should_run_drc	drc	1.0
update_check_op	drc	1.0
update_rule	drc	1.0
adjust_pll_refclk	eco	1.0
create_wirelut	eco	1.0
eco reroute	eco	1.0
eco unload_design	eco	1.0
fitter_report_timing	eco	1.0
fitter_timing_summary	eco	1.0
get_available_snapshots	eco	1.0
get_eco_checkpoint	eco	1.0
get_loaded_snapshot	eco	1.0
get_lutmask_equation	eco	1.0
get_node_location	eco	1.0
make_connection	eco	1.0
modify_io_current_strength	eco	1.0
modify_io_delay_chain	eco	1.0
modify_io_slew_rate	eco	1.0
place_node	eco	1.0
remove_connection	eco	1.0
remove_node	eco	1.0
report_connections	eco	1.0
report_legal_locations	eco	1.0
report_nodes_at_location	eco	1.0
report_ports	eco	1.0
report_routing	eco	1.0
report_unplaced_nodes	eco	1.0
restore_eco_checkpoint	eco	1.0
unplace_node	eco	1.0
update_mif_files	eco	1.0
apply_setting	external_memif_toolkit	1.0
calibrate_termination	external_memif_toolkit	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
configure_driver	external_memif_toolkit	1.0
create_connection_report	external_memif_toolkit	1.0
create_toolkit_report	external_memif_toolkit	1.0
driver_margining	external_memif_toolkit	1.0
establish_connection	external_memif_toolkit	1.0
generate_eye_diagram	external_memif_toolkit	1.0
get_connection_commands	external_memif_toolkit	1.0
get_connection_info	external_memif_toolkit	1.0
get_connection_interfaces	external_memif_toolkit	1.0
get_connection_report_info	external_memif_toolkit	1.0
get_connection_report_types	external_memif_toolkit	1.0
get_connection_types	external_memif_toolkit	1.0
get_connections	external_memif_toolkit	1.0
get_setting_types	external_memif_toolkit	1.0
get_toolkit_report_types	external_memif_toolkit	1.0
initialize_connections	external_memif_toolkit	1.0
link_project_to_device	external_memif_toolkit	1.0
read_setting	external_memif_toolkit	1.0
reindex_connections	external_memif_toolkit	1.0
reset_tg2	external_memif_toolkit	1.0
run_connection_command	external_memif_toolkit	1.0
set_active_interface	external_memif_toolkit	1.0
set_stress_pattern	external_memif_toolkit	1.0
terminate_connection	external_memif_toolkit	1.0
terminate_connections	external_memif_toolkit	1.0
unlink_project_from_device	external_memif_toolkit	1.0
write_connection_target_report	external_memif_toolkit	1.0
check	fif	1.0
dump	fif	1.0
dump_cram_frame	fif	1.0
dump_mem	fif	1.0
dump_pr_bitstream	fif	1.0
generate	fif	1.0
get_frame_count	fif	1.0
get_frame_size	fif	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
get_sector_information_sdm_based_fp ga	fif	1.0
get_sensitive_location	fif	1.0
get_sensitive_location_sdm_based_fpg a	fif	1.0
setup	fif	1.0
setup_sdm_based_fpga	fif	1.0
terminate	fif	1.0
add_object	flng	1.0
add_property	flng	1.0
bind_flow	flng	1.0
delete_object	flng	1.0
get_flow_list	flng	1.0
get_next_available_id	flng	1.0
get_object	flng	1.0
get_objects	flng	1.0
get_option	flng	1.0
get_property	flng	1.0
get_task_command	flng	1.0
init_repository	flng	1.0
list_properties	flng	1.0
monitor_flow	flng	1.0
run_flow	flng	1.0
set_option	flng	1.0
set_property	flng	1.0
execute_flow	flow	1.1
execute_module	flow	1.1
get_flow_templates	flow	1.1
write_flow_assignment_digest	flow	1.1
write_flow_finished	flow	1.1
write_flow_started	flow	1.1
write_flow_template	flow	1.1
help_arg_examples	help	1.0
begin_memory_edit	insystem_memory_edit	1.0
end_memory_edit	insystem_memory_edit	1.0
get_editable_mem_instances	insystem_memory_edit	1.0
read_content_from_memory	insystem_memory_edit	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
save_content_from_memory_to_file	insystem_memory_edit	1.0
update_content_to_memory_from_file	insystem_memory_edit	1.0
write_content_to_memory	insystem_memory_edit	1.0
end_insystem_source_probe	insystem_source_probe	1.0
get_insystem_source_probe_instance_info	insystem_source_probe	1.0
read_probe_data	insystem_source_probe	1.0
read_source_data	insystem_source_probe	1.0
start_insystem_source_probe	insystem_source_probe	1.0
write_source_data	insystem_source_probe	1.0
analyze_files	interactive_synthesis	1.0
check_rtl_connections	interactive_synthesis	1.0
dissolve_rtl_partition	interactive_synthesis	1.0
elaborate	interactive_synthesis	1.0
get_entities	interactive_synthesis	1.0
get_rtl_partition_name	interactive_synthesis	1.0
get_rtl_partitions	interactive_synthesis	1.0
link_rtl_design	interactive_synthesis	1.0
print_ipxact	interactive_synthesis	1.0
report_rtl_assignments	interactive_synthesis	1.0
report_rtl_parameters	interactive_synthesis	1.0
report_rtl_stats	interactive_synthesis	1.0
reset_rtl_design	interactive_synthesis	1.0
save_rtl_design	interactive_synthesis	1.0
synthesize	interactive_synthesis	1.0
uniquify	interactive_synthesis	1.0
write_rtl_report	interactive_synthesis	1.0
clear_ip_generation_dirs	ipgen	1.0
generate_ip_file	ipgen	1.0
generate_project_ip_files	ipgen	1.0
get_project_ip_files	ipgen	1.0
compute_pll	iptclgen	1.0
generate_vhdl_simgen_model	iptclgen	1.0
parse_hdl	iptclgen	1.0
parse_tcl	iptclgen	1.0
close_device	jtag	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
device_dr_shift	jtag	1.0
device_ir_shift	jtag	1.0
device_lock	jtag	1.0
device_run_test_idle	jtag	1.0
device_unlock	jtag	1.0
device_virtual_dr_shift	jtag	1.0
device_virtual_ir_shift	jtag	1.0
get_device_names	jtag	1.0
get_hardware_names	jtag	1.0
open_device	jtag	1.0
begin_logic_analyzer_interface_control	logic_analyzer_interface	1.0
change_bank_to_output_pin	logic_analyzer_interface	1.0
end_logic_analyzer_interface_control	logic_analyzer_interface	1.0
get_current_state_of_output_pin	logic_analyzer_interface	1.0
tristate_output_pin	logic_analyzer_interface	1.0
checksum	misc	1.0
disable_natural_bus_naming	misc	1.0
enable_natural_bus_naming	misc	1.0
escape_brackets	misc	1.0
foreach_in_collection	misc	1.0
get_collection_size	misc	1.0
get_environment_info	misc	1.0
get_message_count	misc	1.0
init_tk	misc	1.0
load	misc	1.0
load_package	misc	1.0
post_message	misc	1.0
qerror	misc	1.0
qexec	misc	1.0
qexit	misc	1.0
record_tcl_cmd	misc	1.0
stopwatch	misc	1.0
get_assignment	names	1.0
set_assignment	names	1.0
initialize	periph	1.0
shutdown	periph	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
check_plan	periph	1.0
get_cell_info	periph	1.0
get_cells	periph	1.0
get_location_info	periph	1.0
get_placement_info	periph	1.0
get_placements	periph	1.0
load_floorplan	periph	1.0
place_cells	periph	1.0
remove_invalid_reports	periph	1.0
report_all	periph	1.0
report_cell_connectivity	periph	1.0
report_cell_placement_reasons	periph	1.0
report_cells	periph	1.0
report_clocks	periph	1.0
report_legal_cell_locations	periph	1.0
report_location_types	periph	1.0
report_locations	periph	1.0
report_regions	periph	1.0
report_summary	periph	1.0
reset_plan	periph	1.0
save_floorplan	periph	1.0
set_clock_type	periph	1.0
undo_last_placement	periph	1.0
unplace_cells	periph	1.0
update_pdpw	periph	1.0
update_plan	periph	1.0
write_plan	periph	1.0
test	pfg	1.0
assignment_group	project	6.0
create_base_clock	project	5.0
create_relative_clock	project	5.0
create_revision	project	7.0
delete_revision	project	7.0
execute_assignment_batch	project	7.0
export_assignments	project	7.0
get_all_assignment_names	project	7.0

*continued...*

Tcl Command	Tcl Package	Package Version
get_all_assignments	project	7.0
get_all_global_assignments	project	7.0
get_all_instance_assignments	project	7.0
get_all_parameters	project	7.0
get_all_quartus_defaults	project	7.0
get_all_user_option_names	project	7.0
get_assignment_info	project	7.0
get_assignment_name_info	project	7.0
get_current_project	project	7.0
get_current_revision	project	7.0
get_database_version	project	7.0
get_global_assignment	project	7.0
get_instance_assignment	project	7.0
get_location_assignment	project	7.0
get_name_info	project	7.0
get_names	project	7.0
get_parameter	project	7.0
get_project_directory	project	7.0
get_project_revisions	project	7.0
get_project_settings	project	1.0
get_top_level_entity	project	7.0
get_user_option	project	7.0
is_database_version_compatible	project	7.0
is_fitter_in_qhd_mode	project	7.0
is_project_open	project	7.0
project_archive	project	7.0
project_clean	project	7.0
project_close	project	7.0
project_exists	project	7.0
project_new	project	7.0
project_open	project	7.0
project_restore	project	7.0
project_settings_exist	project	1.0
remove_all_global_assignments	project	7.0
remove_all_instance_assignments	project	7.0
remove_all_parameters	project	7.0

*continued...*

Tcl Command	Tcl Package	Package Version
resolve_file_path	project	7.0
revision_exists	project	7.0
set_current_revision	project	7.0
set_global_assignment	project	7.0
set_high_effort_fmax_optimization_assignments	project	7.0
set_instance_assignment	project	7.0
set_io_assignment	project	7.0
set_location_assignment	project	7.0
set_multicycle_assignment	project	5.0
set_parameter	project	7.0
set_power_file_assignment	project	7.0
set_project_settings	project	1.0
set_timing_cut_assignment	project	5.0
set_user_option	project	7.0
test_assignment_trait	project	7.0
timegroup	project	4.0
qshm_connect_to_quartus	qshm	1.0
qshm_disconnect_from_quartus	qshm	1.0
qshm_dispose_client	qshm	1.0
qshm_get_hub_key_prefix	qshm	1.0
qshm_get_parent_hub_key	qshm	1.0
qshm_obtain_client	qshm	1.0
qshm_send_request	qshm	1.0
qshm_send_server_state_query	qshm	1.0
qshm_set_context	qshm	1.0
add_row_to_table	report	2.1
create_report_panel	report	2.1
delete_report_panel	report	2.1
get_fitter_resource_usage	report	2.1
get_number_of_columns	report	2.1
get_number_of_rows	report	2.1
get_report_panel_column_index	report	2.1
get_report_panel_data	report	2.1
get_report_panel_id	report	2.1
get_report_panel_names	report	2.1

*continued...*

Tcl Command	Tcl Package	Package Version
get_report_panel_row	report	2.1
get_report_panel_row_index	report	2.1
load_report	report	2.1
read_xml_report	report	2.1
refresh_report_window	report	2.1
save_report_database	report	2.1
unload_report	report	2.1
write_report_panel	report	2.1
write_xml_report	report	2.1
all_clocks	sdc	1.5
all_inputs	sdc	1.5
all_outputs	sdc	1.5
all_registers	sdc	1.5
create_clock	sdc	1.5
create_generated_clock	sdc	1.5
derive_clocks	sdc	1.5
get_cells	sdc	1.5
get_clocks	sdc	1.5
get_nets	sdc	1.5
get_pins	sdc	1.5
get_ports	sdc	1.5
remove_clock_groups	sdc	1.5
remove_clock_latency	sdc	1.5
remove_clock_uncertainty	sdc	1.5
remove_disable_timing	sdc	1.5
remove_input_delay	sdc	1.5
remove_output_delay	sdc	1.5
reset_design	sdc	1.5
set_clock_groups	sdc	1.5
set_clock_latency	sdc	1.5
set_clock_uncertainty	sdc	1.5
set_disable_timing	sdc	1.5
set_false_path	sdc	1.5
set_input_delay	sdc	1.5
set_input_transition	sdc	1.5
set_max_delay	sdc	1.5

*continued...*

Tcl Command	Tcl Package	Package Version
set_max_time_borrow	sdc	1.5
set_min_delay	sdc	1.5
set_multicycle_path	sdc	1.5
set_output_delay	sdc	1.5
derive_clock_uncertainty	sdc_ext	2.0
derive_pll_clocks	sdc_ext	2.0
disable_min_pulse_width	sdc_ext	2.0
get_active_clocks	sdc_ext	2.0
get_assignment_groups	sdc_ext	1.0
get_fanins	sdc_ext	2.0
get_fanouts	sdc_ext	2.0
get_keepers	sdc_ext	2.0
get_nodes	sdc_ext	2.0
get_partitions	sdc_ext	2.0
get_registers	sdc_ext	2.0
remove_annotated_delay	sdc_ext	2.0
remove_clock	sdc_ext	2.0
reset_timing_derate	sdc_ext	2.0
set_active_clocks	sdc_ext	2.0
set_annotated_delay	sdc_ext	2.0
set_data_delay	sdc_ext	2.0
set_max_skew	sdc_ext	2.0
set_net_delay	sdc_ext	2.0
set_scc_mode	sdc_ext	2.0
set_time_format	sdc_ext	2.0
set_timing_derate	sdc_ext	2.0
add_to_collection	sta	1.0
check_timing	sta	1.0
create_report_histogram	sta	1.0
create_slack_histogram	sta	1.0
create_timing_netlist	sta	1.0
create_timing_summary	sta	1.0
delete_sta_collection	sta	1.0
delete_timing_netlist	sta	1.0
enable_ccpp_removal	sta	1.0
enable_sdc_extension_collections	sta	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
get_available_operating_conditions	sta	1.0
get_cell_info	sta	1.0
get_clock_domain_info	sta	1.0
get_clock_fmax_info	sta	1.0
get_clock_info	sta	1.0
get_clock_pair_info	sta	1.0
get_datasheet	sta	1.0
get_default_sdc_file_names	sta	1.0
get_edge_info	sta	1.0
get_entity_instances	sta	1.0
get_min_pulse_width	sta	1.0
get_net_info	sta	1.0
get_node_info	sta	1.0
get_object_info	sta	1.0
get_operating_conditions	sta	1.0
get_operating_conditions_info	sta	1.0
get_partition_info	sta	1.0
get_path	sta	1.0
get_path_info	sta	1.0
get_pin_info	sta	1.0
get_point_info	sta	1.0
get_port_info	sta	1.0
get_register_info	sta	1.0
get_timing_paths	sta	1.0
import_sdc	sta	1.0
locate	sta	1.0
print_total_sdc_processing_time	sta	1.0
query_collection	sta	1.0
read_sdc	sta	1.0
register_delete_timing_netlist_callback	sta	1.0
remove_from_collection	sta	1.0
report_advanced_io_timing	sta	1.0
report_asynch_cdc	sta	1.0
report_bottleneck	sta	1.0
report_cdc_viewer	sta	1.0
report_clock_fmax_summary	sta	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
report_clock_network	sta	1.0
report_clock_transfers	sta	1.0
report_clocks	sta	1.0
report_datasheet	sta	1.0
report_ddr	sta	1.0
report_design_metrics	sta	1.0
report_exceptions	sta	1.0
report_ini_usage	sta	1.0
report_logic_depth	sta	1.0
report_max_clock_skew	sta	1.0
report_max_skew	sta	1.0
report_metastability	sta	1.0
report_min_pulse_width	sta	1.0
report_neighbor_paths	sta	1.0
report_net_delay	sta	1.0
report_net_timing	sta	1.0
report_partitions	sta	1.0
report_path	sta	1.0
report_pipelining_info	sta	1.0
report_register_spread	sta	1.0
report_reset_statistics	sta	1.0
report_retiming_restrictions	sta	1.0
report_route_net_of_interest	sta	1.0
report_rskm	sta	1.0
report_sdc	sta	1.0
report_skew	sta	1.0
report_tccs	sta	1.0
report_timing	sta	1.0
report_timing_by_source_files	sta	1.0
report_timing_tree	sta	1.0
report_ucp	sta	1.0
set_operating_conditions	sta	1.0
timing_netlist_exist	sta	1.0
update_timing_netlist	sta	1.0
use_timing_analyzer_styleEscaping	sta	1.0
write_sdc	sta	1.0

*continued...*

Tcl Command	Tcl Package	Package Version
close_session	stp	1.0
export_data_log	stp	1.0
open_session	stp	1.0
run	stp	1.0
run_multiple_end	stp	1.0
run_multiple_start	stp	1.0
stop	stp	1.0
is_place	tdc	1.0
is_plan	tdc	1.0
is_post_route	tdc	1.0
write_verilog	uno	1.0

### 3.1.1. ::quartus::backannotate

The following table displays information for the **::quartus::backannotate** Tcl package:

<b>Tcl Package and Version</b>	::quartus::backannotate 1.1
<b>Description</b>	This package contains the set of Tcl functions for back-annotating assignments for a project.
<b>Availability</b>	This package is available for loading in the following executables:  qpro quartus quartus_cdb
<b>Tcl Commands</b>	<code>get_back_annotation_assignments</code> ( <b>::quartus::backannotate</b> ) on page 66 <code>logiclock_back_annotate</code> ( <b>::quartus::backannotate</b> ) on page 67

#### 3.1.1.1. get\_back\_annotation\_assignments (::quartus::backannotate)

The following table displays information for the `get_back_annotation_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <b>::quartus::backannotate</b> on page 66	
<b>Syntax</b>	<code>get_back_annotation_assignments [-h   -help] [-long_help]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
<b>Description</b>	Returns an output collection of back-annotation assignments. Each element of the collection is a list with the following format: { {<Source>} {<Destination>} {<Assignment name>} {<Assignment value>} {<Entity name>} }	
<b>Example Usage</b>	<pre>## Print out all the back-annotation assignments set asgn_col [get_back_annotation_assignments] foreach_in_collection asgn \$asgn_col {     ## Each element in the collection has the following     ## format:     ## { &lt;Source&gt;} {&lt;Destination&gt;} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;} {&lt;Entity name&gt;} }</pre>	
	<i>continued...</i>	

	<pre> set from  [lindex \$asgn 0] set to    [lindex \$asgn 1] set name  [lindex \$asgn 2] set value [lindex \$asgn 3] set entity [lindex \$asgn 4] puts "\$entity : \$name (\$from -&gt; \$to) = \$value" } </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Back annotation failed -- design did not compile properly. Run a successful compilation before performing back-annotation.
	TCL_ERROR	1	ERROR: Project has no active revision. Make sure there is an open, active revision.
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: Device or device family does not support node location back annotation.
	TCL_ERROR	1	ERROR: Device or device family does not support LogicLock back annotation.
	TCL_ERROR	1	ERROR: Wrong number of arguments. For correct syntax, refer to help for the logiclock_back_annotate command.

### 3.1.1.2. logiclock\_back\_annotate (::quartus::backannotate)

The following table displays information for the logiclock\_back\_annotate Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::backannotate on page 66	
<b>Syntax</b>	<pre> logiclock_back_annotate [-h   -help] [-long_help] [-exclude_from] [-exclude_to] [-from &lt;source name&gt;] [-lock] [-no_contents] [-no_delay_chain] [-no_demote_lab] [-no_demote_mac] [-no_demote_pin] [-no_demote_ram] [-no_dont_touch] [-path_exclude &lt;path_exclude name&gt;] [-region &lt;region name&gt;] [-remove_assignments] [-resource_filter &lt;resource_filter value&gt;] [-routing] [-to &lt;destination name&gt;] </pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-exclude_from	Option to exclude the source node
	-exclude_to	Option to exclude the destination node
	-from <source name>	Name (or wildcard expression) of the source node to be back-annotated
	-lock	Option to lock back-annotated regions
	-no_contents	Option not to back-annotate contents
	-no_delay_chain	Option not to back-annotate delay chain settings
	-no_demote_lab	Option not to demote LAB or LE assignments
	-no_demote_mac	Option not to demote DSP block assignments
	-no_demote_pin	Option not to demote pin assignments

*continued...*

	<table border="1"> <tr><td>-no_demote_ram</td><td>Option not to demote RAM assignments</td></tr> <tr><td>-no_dont_touch</td><td>Option not to set the don't_touch flag for each back-annotated node</td></tr> <tr><td>-path_exclude &lt;path_exclude name&gt;</td><td>Option to exclude the specified node from the path filter</td></tr> <tr><td>-region &lt;region name&gt;</td><td>Name (or wildcard expression) of region to be back-annotated</td></tr> <tr><td>-remove_assignments</td><td>Option to remove matching assignments instead of creating them</td></tr> <tr><td>-resource_filter &lt;resource_filter value&gt;</td><td>Option to use the resource filter</td></tr> <tr><td>-routing</td><td>Option to back-annotate the LogicLock region's routing</td></tr> <tr><td>-to &lt;destination name&gt;</td><td>Name (or wildcard expression) of the destination node to be back-annotated</td></tr> </table>	-no_demote_ram	Option not to demote RAM assignments	-no_dont_touch	Option not to set the don't_touch flag for each back-annotated node	-path_exclude <path_exclude name>	Option to exclude the specified node from the path filter	-region <region name>	Name (or wildcard expression) of region to be back-annotated	-remove_assignments	Option to remove matching assignments instead of creating them	-resource_filter <resource_filter value>	Option to use the resource filter	-routing	Option to back-annotate the LogicLock region's routing	-to <destination name>	Name (or wildcard expression) of the destination node to be back-annotated
-no_demote_ram	Option not to demote RAM assignments																
-no_dont_touch	Option not to set the don't_touch flag for each back-annotated node																
-path_exclude <path_exclude name>	Option to exclude the specified node from the path filter																
-region <region name>	Name (or wildcard expression) of region to be back-annotated																
-remove_assignments	Option to remove matching assignments instead of creating them																
-resource_filter <resource_filter value>	Option to use the resource filter																
-routing	Option to back-annotate the LogicLock region's routing																
-to <destination name>	Name (or wildcard expression) of the destination node to be back-annotated																
<b>Description</b>	Back-annotates a LogicLock region and its contents. When you use the "-routing" option, you must use the "-lock" and "-no_demote_lab" options, without the "-no_contents" option, or use the"-remove_assignments" option. The "-remove_assignments" option removes all matching region contents. When you use the "-remove_assignments" option, the demotion options, "-no_contents" and "-lock", are not applicable. The "-resource_filter" option allows you to back-annotate only specific resource types on the device. For example: logiclock_back_annotate -resource_filter "COMBINATORIAL" This command back-annotates all combinatorial nodes in the design. The complete set of options is: COMBINATORIAL combinatorial nodes REGISTER registered nodes MEGA M-RAMs MEDIUM M4K memory blocks SMALL M512 memory blocks IO I/O elements MAC DSP blocks Intel recommends that you use a Verilog Quartus(R) Mapping File (.vqm) as the source. When any of the advanced netlist optimizations are enabled, it is possible for the Fitter to create and rename nodes in the design during a place and route operation. Back annotation requires that on subsequent compilations the node names in the netlist match those in the constraint file. Write out a VQM netlist and create a new project using that netlist as its source. Copy all of the existing constraint files into the new project directory and remove all the design files except the new .vqm by using the Add/Remove Files in a Project command (Project menu) in the Quartus Prime GUI. The Quartus Prime software will create a root region if you back-annotate nodes that are not members of a LogicLock region. The root region is device-size and locked. You can make assignments to the root region but you cannot delete it or modify its size or location.																
<b>Example Usage</b>	<pre># Open the project "example_project" project_open example_project  # Compile the design package require ::quartus::flow execute_flow -compile  package require ::quartus::backannotate  # Back annotate all nodes and routing in the region "one_region" logiclock_back_annotate -routing -lock -no_demote_lab -region one_region  # Back annotate the location of the nodes on all paths that # start with a node that matches the "Data_in*" wildcard # expression, and end with a node that matches the "Data_out*" # wildcard expression logiclock_back_annotate -from Data_in* -to Data_out*  # Back annotate the placement of all the registers in the design logiclock_back_annotate -resource_filter "REGISTER"  # Close the project project_close</pre>																
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Back annotation failed -- design did not compile properly. Run a successful compilation before performing back-annotation.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Back annotation failed -- design did not compile properly. Run a successful compilation before performing back-annotation.							
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>															
TCL_OK	0	INFO: Operation successful															
TCL_ERROR	1	ERROR: Back annotation failed -- design did not compile properly. Run a successful compilation before performing back-annotation.															

*continued...*

TCL_ERROR	1	ERROR: Project has no active revision. Make sure there is an open, active revision.
TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
TCL_ERROR	1	ERROR: Device or device family does not support node location back annotation.
TCL_ERROR	1	ERROR: The -routing option is used with incompatible options. To use the -routing option, you must use the -lock and -no_demote_lab options without the -no_contents option, or use the -remove_assignments option.
TCL_ERROR	1	ERROR: Device or device family does not support LogicLock back annotation.
TCL_ERROR	1	ERROR: Wrong number of arguments. For correct syntax, refer to help for the logiclock_back_annotate command.

### 3.1.2. ::quartus::bpps

The following table displays information for the **::quartus::bpps** Tcl package:

<b>Tcl Package and Version</b>	::quartus::bpps 1.0
<b>Description</b>	This package provides non-backend support for pin-planner mode in Interface Planner.
<b>Availability</b>	<p>This package is loaded by default in the following executables:</p> <pre>qacv qappl qpro quartus quartus_bpps quartus_da quartus_drc quartus_pdp quartus_pow quartus_sta quartus_staw</pre>
<b>Tcl Commands</b>	<p><a href="#">bpps::apply_assignments</a> (::quartus::bpps) on page 70  <a href="#">bpps::check_plan</a> (::quartus::bpps) on page 70  <a href="#">bpps::export_constraints_to_gsf</a> (::quartus::bpps) on page 71  <a href="#">bpps::get_cell_info</a> (::quartus::bpps) on page 71  <a href="#">bpps::get_device</a> (::quartus::bpps) on page 72  <a href="#">bpps::get_hdbpath_from_id</a> (::quartus::bpps) on page 72  <a href="#">bpps::get_id_from_hdbpath</a> (::quartus::bpps) on page 73  <a href="#">bpps::get_location_info</a> (::quartus::bpps) on page 74  <a href="#">bpps::get_placement</a> (::quartus::bpps) on page 74  <a href="#">bpps::get_placement_info</a> (::quartus::bpps) on page 75  <a href="#">bpps::get_placements</a> (::quartus::bpps) on page 75  <a href="#">bpps::get_placements_of_group</a> (::quartus::bpps) on page 76  <a href="#">bpps::harden_cell</a> (::quartus::bpps) on page 76  <a href="#">bpps::harden_cells</a> (::quartus::bpps) on page 77  <a href="#">bpps::initialize</a> (::quartus::bpps) on page 77  <a href="#">bpps::load_floorplan</a> (::quartus::bpps) on page 78  <a href="#">bpps::place_cells</a> (::quartus::bpps) on page 78  <a href="#">bpps::read_tpl_placement</a> (::quartus::bpps) on page 79  <a href="#">bpps::remove_invalid_reports</a> (::quartus::bpps) on page 80  <a href="#">bpps::report_all</a> (::quartus::bpps) on page 80  <a href="#">bpps::report_cell_connectivity</a> (::quartus::bpps) on page 80  <a href="#">bpps::report_cell_placement_reasons</a> (::quartus::bpps) on page 81  <a href="#">bpps::report_cells</a> (::quartus::bpps) on page 81  <a href="#">bpps::report_clocks</a> (::quartus::bpps) on page 82  <a href="#">bpps::report_legal_cell_locations</a> (::quartus::bpps) on page 82  <a href="#">bpps::report_location_types</a> (::quartus::bpps) on page 83  <a href="#">bpps::report_locations</a> (::quartus::bpps) on page 83  <a href="#">bpps::report_regions</a> (::quartus::bpps) on page 83  <a href="#">bpps::report_summary</a> (::quartus::bpps) on page 84  <a href="#">bpps::reset_plan</a> (::quartus::bpps) on page 84  <a href="#">bpps::save_floorplan</a> (::quartus::bpps) on page 85  <a href="#">bpps::save_pin_assignments</a> (::quartus::bpps) on page 85  <a href="#">bpps::set_mode</a> (::quartus::bpps) on page 86  <a href="#">bpps::shutdown</a> (::quartus::bpps) on page 86  <a href="#">bpps::soften_cell</a> (::quartus::bpps) on page 86  <a href="#">bpps::soften_cells</a> (::quartus::bpps) on page 87</p>

*continued...*

	<b>bpps::undo_last_placement</b> (::quartus::bpsps) on page 87 <b>bpps::unplace_cells</b> (::quartus::bpsps) on page 88 <b>bpps::update_pdpw</b> (::quartus::bpsps) on page 88 <b>bpps::validate_placement</b> (::quartus::bpsps) on page 89 <b>bpps::write_plan</b> (::quartus::bpsps) on page 89 <b>bpps::write_tpl_placement</b> (::quartus::bpsps) on page 90
--	--

### 3.1.2.1. bpps::apply\_assignments (::quartus::bpsps)

The following table displays information for the `bpsps::apply_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bpsps</a> on page 69		
<b>Syntax</b>	<code>bpsps::apply_assignments [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	In classic mode, applies all changes to constraints and reloads them into Interface Planner. After the platform has been updated with the constraints, placement operations can be performed. In pin planner mode, loads the QSF constraints related to pin assignments.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsps::update_plan blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.2. bpps::check\_plan (::quartus::bpsps)

The following table displays information for the `bpsps::check_plan` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bpsps</a> on page 69		
<b>Syntax</b>	<code>bpsps::check_plan [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	In classic mode, checks the legality of the current plan. In pin planner mode, this will be a stub. Assignments are checked real time, no backend engine to check legality of anything.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsps::update_plan bpsps::check_plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.3. bpps::export\_constraints\_to\_qsf (::quartus::bps)

The following table displays information for the bpps::export\_constraints\_to\_qsf Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::export_constraints_to_qsf [-h   -help] [-long_help] [-bb_locations] [-close_pdp] [-disabled] [-pin_locations] [-tile_locations]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-bb_locations	Write out building block location assignments	
	-close_pdp	Send call back to PDPW to close after exporting is done	
	-disabled	Write out disabled assignments	
	-pin_locations	Write out pin location assignments	
	-tile_locations	Write out tile location assignments	
<b>Description</b>	In Tile Planner mode, export constraints to qsf file		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan set io_cells [bps::get_cells -unplaced -type IO_CLUSTER] bps::place_cells -cells \$io_cells bps::validate_placement bps::export_constraints -disabled -tile_locations -bb_locations project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.4. bpps::get\_cell\_info (::quartus::bps)

The following table displays information for the bpps::get\_cell\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::get_cell_info [-h   -help] [-long_help] [-children] [-guide_cell_id] [-ip_type] [-links] [-location] [-name] [-parent] [-type] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-children	Return the the cell id of the children cells	
	-guide_cell_id	Returns the guide cell's elem_id	
	-ip_type	Returns the IP type if the cell is an IP cell or an empty string otherwise	
	-links	Return the given design element's connections to other cells	
	<i>continued...</i>		

	-location	Returns the location ID if the cell is placed or an empty string otherwise	
	-name	Return the cell name of the cell id	
	-parent	Return the the cell id of the parent cells	
	-type	Return the the type of the cell	
	<cell_id>	Single cell id	
<b>Description</b>	Gets information about the specified cell (referenced by cell ID). You can obtain cell using the periph::get_cells Tcl command.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan  foreach cell [periph::get_cells -type IO_CLUSTER] {     puts "Found cell ID \$cell named [periph::get_cell_info -name \$cell]" }  blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.5. bpps::get\_device (::quartus::bps)

The following table displays information for the bpps::get\_device Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::get_device [-h   -help] [-long_help] [-compress]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-compress	Compress requested data	
<b>Description</b>	Internal function to get the device tree in json, this gets the complete device model.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.6. bpps::get\_hdbpath\_from\_id (::quartus::bps)

The following table displays information for the bpps::get\_hdbpath\_from\_id Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::get_hdbpath_from_id [-h   -help] [-long_help] [-design_cell_id <design_cell_id>] [-device_loc_id <device_loc_id> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	

*continued...*

	-design_cell_id <design_cell_id>	design cell ID	
	-device_loc_id <device_loc_id>	device location ID	
<b>Description</b>	In classic mode, load the floorplan from a Interface Planner floorplan file In pin planner mode, this is a stub.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpps::update_plan bpps::place_cells -unplaced_cells bpps::save_floorplan -filename onewire_blueprint_floorplan.plan bpps::load_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.7. bpps::get\_id\_from\_hdbpath (::quartus::bps)

The following table displays information for the bpps::get\_id\_from\_hdbpath Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bpps::get_id_from_hdbpath [-h   -help] [-long_help] [-design_cell <design_cell> ] [-device_loc <device_loc> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-design_cell <design_cell>	HDB Path of design cell	
	-device_loc <device_loc>	HDB Path of device location	
<b>Description</b>	In classic mode, load the floorplan from a Interface Planner floorplan file In pin planner mode, this is a stub.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpps::update_plan bpps::place_cells -unplaced_cells bpps::save_floorplan -filename onewire_blueprint_floorplan.plan bpps::load_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.8. **bpps::get\_location\_info** (::quartus::bps)

The following table displays information for the `bpps::get_location_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bps</a> on page 69		
<b>Syntax</b>	<code>bpps::get_location_info [-h   -help] [-long_help] [-children] [-gid] [-name] [-parents] [-placed_cells] [-properties] [-type] &lt;location_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-children</code>	Query the children location IDs	
	<code>-gid</code>	Query the gid of the location IDs	
	<code>-name</code>	Return the location name of the location id	
	<code>-parents</code>	Query the parent location IDs	
	<code>-placed_cells</code>	Return the placed cells at the location id	
	<code>-properties</code>	Return the device location properties in json	
	<code>-type</code>	Return the location type of the location id	
	<code>&lt;location_id&gt;</code>	location id	
<b>Description</b>	Gets information about the specified location (referenced by location ID). You can obtain location using the <code>periph::get_locations</code> Tcl command or using the <code>bpps::get_location_info -properties &lt;loc_id&gt;</code> Tcl command		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan  foreach cell [periph::get_cells -placed] {     puts "Found cell ID \$cell named [periph::get_cell_info -name \$cell] placed in location [periph::get_cell_info -location \$cell] named [periph::get_location_info -name [periph::get_cell_info -location \$cell]]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.9. **bpps::get\_placement** (::quartus::bps)

The following table displays information for the `bpps::get_placement` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bps</a> on page 69		
<b>Syntax</b>	<code>bpps::get_placement [-h   -help] [-long_help] [-compress]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-compress</code>	Compress requested data	
<b>Description</b>	Return information about design placement		
<i>continued...</i>			

<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

### 3.1.2.10. bpps::get\_placement\_info (::quartus::bps)

The following table displays information for the `bpsp::get_placement_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpsp::get_placement_info [-h   -help] [-long_help] [-placement] &lt;placement_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-placement</code>	Return the placement as a list of cell/id pairs	
	<code>&lt;placement_id&gt;</code>	Single placement id	
<b>Description</b>	In classic mode, return information about a given placement In pin planner mode, looking up placement objects is not supported, no such thing.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

### 3.1.2.11. bpps::get\_placements (::quartus::bps)

The following table displays information for the `bpsp::get_placements` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpsp::get_placements [-h   -help] [-long_help] [-ips] &lt;cell_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-ips</code>	Get placements for IPs only	
	<code>&lt;cell_id&gt;</code>	Single cell id	
<b>Description</b>	In classic mode, returns a vector of placements for the supplied cell In pin planner mode, returns a vector of compatible locations for the supplied pin		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

**continued...**

TCL_ERROR	1	ERROR: At least one pin ID must be supplied, but no IDs were supplied
TCL_ERROR	1	ERROR: The supplied pin id <string> is not a placeable pin.
TCL_ERROR	1	ERROR: <string> IDs expected, but <string> were supplied

### 3.1.2.12. bpps::get\_placements\_of\_group (::quartus::bps)

The following table displays information for the bpps::get\_placements\_of\_group Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bpsp::get_placements_of_group [-h   -help] [-long_help] -cells <cells> [-ips]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cells <cells>	One or more cell ids	
	-ips	Get placements for IPs only	
<b>Description</b>	Given a list of design cell IDs, returns a vector of possible placement IDs for all the cells.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsp::update_plan bpsp::place_cells -unplaced_cells bpsp::check_plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied ID <string> is invalid.
	TCL_ERROR	1	ERROR: At least one ID must be supplied, but no IDs were supplied
	TCL_ERROR	1	ERROR: <string> IDs expected, but <string> were supplied

### 3.1.2.13. bpps::harden\_cell (::quartus::bps)

The following table displays information for the bpps::harden\_cell Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bpsp::harden_cell [-h   -help] [-long_help] -cell <cell>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cell <cell>	Harden the existing placement of the specified cell	

*continued...*

<b>Description</b>	In modes that support soft / hard placements (ie. Tile Planner mode), hardens the existing placement of the specified cell. If the cell is subsequently unplaced and placed again, the placement soft / hard attribute will be based on the new placement action. For modes that do not support soft / hard placements, nothing is performed.		
<b>Example Usage</b>	<code>bpps::harden_cell -cell &lt;design_cell_id&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.14. bpps::harden\_cells (::quartus::bps)

The following table displays information for the `bpps::harden_cells` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::harden_cells [-h   -help] [-long_help] -cells &lt;cells&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cells <cells>	Harden the existing placement of the specified cells	
<b>Description</b>	In modes that support soft / hard placements (ie. Tile Planner mode), hardens the existing placement of the specified cell. If the cell is subsequently unplaced and placed again, the placement soft / hard attribute will be based on the new placement action. For modes that do not support soft / hard placements, nothing is performed.		
<b>Example Usage</b>	<code>bpps::harden_cells -cells [&lt;design_cell_id&gt;]</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.15. bpps::initialize (::quartus::bps)

The following table displays information for the `bpps::initialize` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::initialize [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Replaces blueprint::initialize command. It will create the design and device models without a backend separate-exe engine.		
<b>Example Usage</b>	<pre>project_open onewire_nf bpps::initialize bpps::update_plan bpps::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.16. bpps::load\_floorplan (::quartus::bps)

The following table displays information for the bpps::load\_floorplan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::load_floorplan [-h   -help] [-long_help] -filename <filename>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-filename <filename>	Filename to load	
<b>Description</b>	In classic mode, load the floorplan from a Interface Planner floorplan file In pin planner mode, this is a stub.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan bps::place_cells -unplaced_cells bps::save_floorplan -filename onewire_blueprint_floorplan.plan bps::load_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.17. bpps::place\_cells (::quartus::bps)

The following table displays information for the bpps::place\_cells Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::place_cells [-h   -help] [-long_help] [-cell_location <cell_location>] [-cells <cells>] [-dont_revert_on_fail] [-fixed_cells] [-placement <placement>] [-unplaced_cells]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cell_location <cell_location>	Cell location id pair to place cells into	
	-cells <cells>	One or more cell ids	
	-dont_revert_on_fail	Option to specify that the best partial placement should be saved on the undo stack upon a placement failure	
	-fixed_cells	Place all unplaced cells	
	-placement <placement>	Place cells according to a placement. A placement is a special object that comes from the bpps::get_placements Tcl command	
	-unplaced_cells	Place all unplaced cells	
<b>Description</b>	In classic mode, performs a placement on the supplied cells In pin planner mode, auto assigns all the pins in the design if possible. Only -cell_ids <cell_id_list> and -cell_location <(cell_id, loc_id)> are actually used. If, exclusively, one of these are not specified, the command does NOTHING.		
<i>continued...</i>			

<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpps::update_plan bpps::place_cells -unplaced_cells bpps::check_plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied ID <string> is invalid.
	TCL_ERROR	1	ERROR: At least one ID must be supplied, but no IDs were supplied
	TCL_ERROR	1	ERROR: <string> IDs expected, but <string> were supplied

### 3.1.2.18. bpps::read\_tpl\_placement (::quartus::bps)

The following table displays information for the bpps::read\_tpl\_placement Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::read_tpl_placement [-h   -help] [-long_help] -filename <filename>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-filename <filename>		Filename to write to
<b>Description</b>	In TilePlanner mode, read placement from a JSON file. Nothing happens (should not be available in GUI) in other modes.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpps::update_plan bpps::place_cells -unplaced_cells bpps::check_plan bpps::write_tpl_placement -filename onewire_blueprint_assignments.tcl project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Filename provided incorrectly

### 3.1.2.19. bpps::remove\_invalid\_reports (::quartus::bps)

The following table displays information for the bpps::remove\_invalid\_reports Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::remove_invalid_reports [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	In classic mode, remove all invalid report In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.20. bpps::report\_all (::quartus::bps)

The following table displays information for the bpps::report\_all Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_all [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	In classic mode, create all default summary reports. In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.21. bpps::report\_cell\_connectivity (::quartus::bps)

The following table displays information for the bpps::report\_cell\_connectivity Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_cell_connectivity [-h   -help] [-long_help] [-fanins] [-fanouts] [-panel_name <name>] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fanins	Report only the fanins of the cell	
	-fanouts	Report only the fanouts of the cell	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	

*continued...*

	<code>&lt;cell_id&gt;</code>	Single cell id
<b>Description</b>	In classic mode, creates a report of the connectivity for a cell. In pin planner mode, this is a stub call.	
<b>Example Usage</b>	This command currently contains no example usage.	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
		INFO: Operation successful

### 3.1.2.22. `bpps::report_cell_placement_reasons` (::quartus::bps)

The following table displays information for the `bpps::report_cell_placement_reasons` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::report_cell_placement_reasons [-h   -help] [-long_help] [-panel_name &lt;name&gt;] &lt;cell_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel	
	<code>&lt;cell_id&gt;</code>	Single cell id	
<b>Description</b>	In classic mode, creates a report of all the locations a particular cell can be placed and the reasons it cannot be placed there. In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.23. `bpps::report_cells` (::quartus::bps)

The following table displays information for the `bpps::report_cells` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::report_cells [-h   -help] [-long_help] [-name &lt;name&gt;] [-panel_name &lt;name&gt;] [-placed] [-type &lt;type&gt;] [-unplaced]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-name &lt;name&gt;</code>	Filter the list of placed cells specifying a name. Wildcards are supported.	
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel	
	<code>-placed</code>	Report the list of placed cells	
	<code>-type &lt;type&gt;</code>	Filter the list of placed cells specifying a list of types	
	<code>-unplaced</code>	Report the list of unplaced cells	

*continued...*

<b>Description</b>	In classic mode, returns a list of periphery cells based on the specified criteria. In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.24. `bpps::report_clocks` (`::quartus::bps`)

The following table displays information for the `bpps::report_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bps</a> on page 69				
<b>Syntax</b>	<code>bpps::report_clocks [-h   -help] [-long_help] [-panel_name &lt;name&gt; ]</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel			
<b>Description</b>	In classic mode, show the signals that are using low-skew routing networks (clock networks) in the device. If applicable, also show any signals that were considered for automatic clock network promotion, but were not promoted. In pin planner mode, this is a stub call.				
<b>Example Usage</b>	This command currently contains no example usage.				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.2.25. `bpps::report_legal_cell_locations` (`::quartus::bps`)

The following table displays information for the `bpps::report_legal_cell_locations` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bps</a> on page 69				
<b>Syntax</b>	<code>bpps::report_legal_cell_locations [-h   -help] [-long_help] [-panel_name &lt;name&gt; ] &lt;cell_id&gt;</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel			
	<cell_id>	Single cell id			
<b>Description</b>	In classic mode, creates a report of the legal periphery cell locations of a cell In pin planner mode, this is a stub call.				
<b>Example Usage</b>	This command currently contains no example usage.				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.2.26. bpps::report\_location\_types (::quartus::bps)

The following table displays information for the bpps::report\_location\_types Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_location_types [-h   -help] [-long_help] [-panel_name <name> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	In classic mode, creates a report of the location types in the periphery In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.27. bpps::report\_locations (::quartus::bps)

The following table displays information for the bpps::report\_locations Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_locations [-h   -help] [-long_help] [-panel_name <name> ] <type>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	<type>	location type to query	
<b>Description</b>	In classic mode, Creates a report of the locations for the requested type in the periphery In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.28. bpps::report\_regions (::quartus::bps)

The following table displays information for the bpps::report\_regions Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_regions [-h   -help] [-long_help] [-panel_name <name> ]		
<b>Arguments</b>	-h   -help	Short help	

*continued...*

	-long_help	Long help with examples and possible return values
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
<b>Description</b>	In pin planner mode, this is a stub call.	
<b>Example Usage</b>	This command currently contains no example usage.	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0

### 3.1.2.29. bpps::report\_summary (::quartus::bps)

The following table displays information for the bpps::report\_summary Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::report_summary [-h   -help] [-long_help] [-panel_name <name> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	In pin planner mode, this is a stub call.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.30. bpps::reset\_plan (::quartus::bps)

The following table displays information for the bpps::reset\_plan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::reset_plan [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	In classic mode, reverts the current design to be unplaced and without assignments applied. In pin planner mode, removes all the user created pin assignments. Keeps the original assignments. (currently just a stub still)		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan bps::reset_plan</pre>		

*continued...*

	<pre>blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.31. bpps::save\_floorplan (::quartus::bps)

The following table displays information for the bpps::save\_floorplan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::save_floorplan [-h   -help] [-long_help] -filename <filename>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-filename <filename>	Filename to write to	
<b>Description</b>	In classic mode, write the Interface Planner floorplan that can be reloaded in Interface Planner In pin planner mode, write the user pin assignments as constraints to QSF file. It is preferred to use the new save_pin_assignments call instead in pin planner mode.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan set io_cells [bps::get_cells -unplaced -type IO_CLUSTER] bps::place_cells -cells \$io_cells bps::save_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.32. bpps::save\_pin\_assignments (::quartus::bps)

The following table displays information for the bpps::save\_pin\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::save_pin_assignments [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Write the Interface Planner floorplan that can be reloaded in Interface Planner		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan</pre>		

*continued...*

	<pre>bpps::save_pin_assignments project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.33. bpps::set\_mode (::quartus::bps)

The following table displays information for the bpps::set\_mode Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::set_mode [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Internal use only for PDPW to set the mode of the middleware. Also defines what plugins will be loaded		
<b>Example Usage</b>	DO NOT call this explicitly		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.34. bpps::shutdown (::quartus::bps)

The following table displays information for the bpps::shutdown Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::shutdown [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Shutdown Interface Planner.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.35. bpps::soften\_cell (::quartus::bps)

The following table displays information for the bpps::soften\_cell Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	bps::soften_cell [-h   -help] [-long_help] -cell <cell>		
<i>continued...</i>			

<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-cell <cell>	Softens the existing placement of the specified cell
<b>Description</b>	In modes that support soft / hard placements (ie. Tile Planner mode), softens the existing placement of the specified cell. If the cell is subsequently unplaced and placed again, the placement soft / hard attribute will be based on the new placement action. For modes that do not support soft / hard placements, nothing is performed.	
	<b>Example Usage</b> <code>bpps::soften_cell -cell &lt;design_cell_id&gt;</code>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0 INFO: Operation successful

### 3.1.2.36. bpps::soften\_cells (::quartus::bps)

The following table displays information for the `bpps::soften_cells` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::soften_cells [-h   -help] [-long_help] [-cells &lt;cells&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cells <cells>	One or more cell ids to soften	
<b>Description</b>	In modes that support soft / hard placements (ie. Tile Planner mode), softens the existing placement of the specified cell. If the cell is subsequently unplaced and placed again, the placement soft / hard attribute will be based on the new placement action. For modes that do not support soft / hard placements, nothing is performed.		
<b>Example Usage</b>	<code>bpps::soften_cells -cells [&lt;design_cell_ids&gt;]</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.37. bpps::undo\_last\_placement (::quartus::bps)

The following table displays information for the `bpps::undo_last_placement` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::undo_last_placement [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	In classic mode, undo the last placement or unplacement operation. In classic mode, undo the last pin assignment or assignment removal operation.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.38. bpps::unplace\_cells (::quartus::bps)

The following table displays information for the bpps::unplace\_cells Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69				
<b>Syntax</b>	bpsps::unplace_cells [-h   -help] [-long_help] [-cells <cells>] [-placed_cells]				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-cells <cells>	One or more cell ids			
	-placed_cells	Unplace all placed cells			
<b>Description</b>	In classic mode, removes the placement from the specified cells. Any constraints for the cells remain, but the cell no longer has a placement. In pin planner mode, similar to classic mode, unassigns pin locations made within this session.				
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsps::update_plan bpsps::unplace_cells -placed_cells bpsps::check_plan project_close</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.2.39. bpps::update\_pdpw (::quartus::bps)

The following table displays information for the bpps::update\_pdpw Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69				
<b>Syntax</b>	bpsps::update_pdpw [-h   -help] [-long_help] [-assignments] [-placement]				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-assignments	Indicates assignment model needs updating			
	-placement	Indicates placement needs updating			
<b>Description</b>	In classic mode, this command update everything that needs updating in pdpw. This essentially sends a single TCL command to pdpw to update everything as needed. Used in the TCL proc source wrapper only. In pin planner mode, this command is just a stub.				
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsps::update_pdpw -pdp_state [blueprint_internal::get_pdp_state]</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.2.40. **bpps::validate\_placement** (::quartus::bps)

The following table displays information for the `bpps::validate_placement` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::validate_placement [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Removes the exception to ignore the given project assignments. The result is the project assignments will take affect on the active design.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.41. **bpps::write\_plan** (::quartus::bps)

The following table displays information for the `bpps::write_plan` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::bps on page 69		
<b>Syntax</b>	<code>bpps::write_plan [-h   -help] [-long_help] [-clocks] [-disabled] -filename &lt;filename&gt; [-force] [-other_locations] [-pin_locations]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-clocks	Write out clock assignments	
	-disabled	Write out disabled assignments	
	-filename <filename>	Filename to write to	
	-force	Force the creation of the plan	
	-other_locations	Write out other location assignments	
	-pin_locations	Write out pin location assignments	
<b>Description</b>	In classic mode, export the floorplan constraints Tcl script In pin planner mode, does nothing (we're not exporting to any TCL file, instead we write to QSF directly in save_floorplan, or save_pin_assignments (recommended), calls.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bps::update_plan bps::place_cells -unplaced_cells bps::check_plan bps::write_plan -filename onewire_blueprint_assignments.tcl project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.2.42. **bpps::write\_tpl\_placement** (::quartus::bps)

The following table displays information for the `bpps::write_tpl_placement` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::bps</a> on page 69		
<b>Syntax</b>	<code>bpps::write_tpl_placement [-h   -help] [-long_help] -filename &lt;filename&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-filename <filename>	Filename to write to	
<b>Description</b>	In TilePlanner mode, write out the placement JSON file. Nothing happens (should not be available in GUI) in other modes.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize bpsp::update_plan bpsp::place_cells -unplaced_cells bpsp::check_plan bpsp::write_tpl_placement -filename onewire_blueprint_assignments.tcl project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Filename provided incorrectly

### 3.1.3. ::quartus::chip\_planner

The following table displays information for the `::quartus::chip_planner` Tcl package:

<b>Tcl Package and Version</b>	::quartus::chip_planner 2.0
<b>Description</b>	This package contains the set of Tcl functions for identifying and modifying resource usage and routing with the Chip Planner.
<b>Availability</b>	This package is available for loading in the following executables:  qacv qpro quartus quartus_cdb
<b>Tcl Commands</b>	<code>check_node</code> ( <a href="#">::quartus::chip_planner</a> ) on page 91 <code>close_chip_planner</code> ( <a href="#">::quartus::chip_planner</a> ) on page 91 <code>design_has_ace_support</code> ( <a href="#">::quartus::chip_planner</a> ) on page 92 <code>design_has_encrypted_ip</code> ( <a href="#">::quartus::chip_planner</a> ) on page 92 <code>get_info_parameters</code> ( <a href="#">::quartus::chip_planner</a> ) on page 92 <code>get_iports</code> ( <a href="#">::quartus::chip_planner</a> ) on page 93 <code>get_node_by_name</code> ( <a href="#">::quartus::chip_planner</a> ) on page 94 <code>get_node_info</code> ( <a href="#">::quartus::chip_planner</a> ) on page 0 <code>get_nodes</code> ( <a href="#">::quartus::chip_planner</a> ) on page 0 <code>get_oports</code> ( <a href="#">::quartus::chip_planner</a> ) on page 94 <code>get_port_by_type</code> ( <a href="#">::quartus::chip_planner</a> ) on page 95 <code>get_port_info</code> ( <a href="#">::quartus::chip_planner</a> ) on page 0 <code>get_sp_pin_list</code> ( <a href="#">::quartus::chip_planner</a> ) on page 96 <code>get_tile_power_setting</code> ( <a href="#">::quartus::chip_planner</a> ) on page 96 <code>read_netlist</code> ( <a href="#">::quartus::chip_planner</a> ) on page 96 <code>set_batch_mode</code> ( <a href="#">::quartus::chip_planner</a> ) on page 97

### 3.1.3.1. check\_node (::quartus::chip\_planner)

The following table displays information for the `check_node` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::chip_planner</code> on page 90				
<b>Syntax</b>	<code>check_node [-h   -help] [-long_help] [-gen_id &lt;gen id&gt;] [-node &lt;node id&gt;]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-gen_id &lt;gen id&gt;</code>	Node generic ID			
	<code>-node &lt;node id&gt;</code>	Node ID			
<b>Description</b>	Checks whether the specified node is legal. Returns 1, if the node is legal. Returns 0, otherwise. Even if a node is legal, you still must run the <code>check_netlist_and_save</code> command to verify the node legality within the netlist.				
<b>Example Usage</b>	<code>check_node -node 3</code>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Conflicting arguments. Consult help for the Tcl command for details.		
	TCL_ERROR	1	ERROR: Illegal node generic ID: %u. Specify a legal node generic ID.		
	TCL_ERROR	1	ERROR: Illegal node ID: %u. Specify a legal node ID.		
	TCL_ERROR	1	ERROR: The node you specified is a legalization node. Modification of legalization nodes is not supported.		
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the <code>read_netlist</code> command.		

### 3.1.3.2. close\_chip\_planner (::quartus::chip\_planner)

The following table displays information for the `close_chip_planner` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::chip_planner</code> on page 90				
<b>Syntax</b>	<code>close_chip_planner [-h   -help] [-long_help]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
<b>Description</b>	Releases the chip planner netlist from use.				
<b>Example Usage</b>	<code>close_chip_planner</code>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.3.3. design\_has\_ace\_support (::quartus::chip\_planner)

The following table displays information for the design\_has\_ace\_support Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	design_has_ace_support [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Determines whether Chip Planner operations can be performed on the current design.		
<b>Example Usage</b>	design_has_ace_support		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.4. design\_has\_encrypted\_ip (::quartus::chip\_planner)

The following table displays information for the design\_has\_encrypted\_ip Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	design_has_encrypted_ip [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Determines whether the current design contains encrypted IP. Returns 1, if the design contains encrypted IP. You may be able to view or edit individual nodes of the design if they are not part of an encrypted IP. To check individual nodes, use the command "get_node_info -node <node id> -info encrypted". Returns 0, otherwise.		
<b>Example Usage</b>	design_has_encrypted_ip		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.5. get\_info\_parameters (::quartus::chip\_planner)

The following table displays information for the get\_info\_parameters Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	get_info_parameters [-h   -help] [-long_help] [-file <file name>] [-for_chip]		
<b>Arguments</b>	-h   -help	Short help	
	<i>continued...</i>		

	-long_help	Long help with examples and possible return values	
	-file <file name>	Name of output file	
	-for_chip	Option to display all of the chip info parameters	
<b>Description</b>	Returns a Tcl list of information parameters. When you use the -file option, the list is redirected to the specified output file. If the output file already exists, it is overwritten without warning.		
<b>Example Usage</b>	<pre>get_info_parameters get_info_parameters -file the_list.txt</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.3.6. get\_iports (::quartus::chip\_planner)

The following table displays information for the get\_iports Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<pre>get_iports [-h   -help] [-long_help] [-as_gen_id] [-gen_id &lt;gen id&gt;] [-node &lt;node id&gt;] [-src_gen_id &lt;gen id&gt;]</pre>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-as_gen_id	Option to return results as generic ID	
	-gen_id <gen id>	Node generic ID	
	-node <node id>	Node id	
	-src_gen_id <gen id>	Source port generic ID	
<b>Description</b>	Returns a collection of input ports for the specified node. You can use the collection with the "foreach_in_collection" command.		
<b>Example Usage</b>	<pre>get_iports -node 3 get_iports -src_gen_id 5</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Conflicting arguments. Consult help for the Tcl command for details.
	TCL_ERROR	1	ERROR: Illegal node generic ID: %u. Specify a legal node generic ID.
	TCL_ERROR	1	ERROR: Illegal node ID: %u. Specify a legal node ID.
	TCL_ERROR	1	ERROR: Illegal oport generic ID: %u. Specify a legal oport generic ID.
	TCL_ERROR	1	ERROR: The node you specified is a legalization node. Modification of legalization nodes is not supported.
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.7. `get_node_by_name` (::quartus::chip\_planner)

The following table displays information for the `get_node_by_name` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>get_node_by_name [-h   -help] [-long_help] [-as_gen_id] -name &lt;node name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-as_gen_id	Option to return result as generic id	
	-name <node name>	Node name	
<b>Description</b>	Returns the node id of the specified node. Returns -1 if the node cannot be found.		
<b>Example Usage</b>	<code>get_node_by_name -name 3</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.8. `get_oports` (::quartus::chip\_planner)

The following table displays information for the `get_oports` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>get_oports [-h   -help] [-long_help] [-as_gen_id] [-gen_id &lt;gen id&gt;] [-node &lt;node id&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-as_gen_id	Option to return results as generic id	
	-gen_id <gen id>	Node generic id	
	-node <node id>	Node id	
<b>Description</b>	Returns a collection of output ports for the specified node. You can use the collection with the foreach_in_collection command.		
<b>Example Usage</b>	<code>get_oports -node 3</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Conflicting arguments. Consult help for the Tcl command for details.
	TCL_ERROR	1	ERROR: Illegal node generic ID: %u. Specify a legal node generic ID.

*continued...*

TCL_ERROR	1	ERROR: Illegal node ID: %u. Specify a legal node ID.
TCL_ERROR	1	ERROR: The node you specified is a legalization node. Modification of legalization nodes is not supported.
TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.9. get\_port\_by\_type (::quartus::chip\_planner)

The following table displays information for the get\_port\_by\_type Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	get_port_by_type [-h   -help] [-long_help] [-as_gen_id] [-gen_id <gen id>] [-literal_index <literal index>] [-node <node id>] -port_type <port type> -type <iport oport>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-as_gen_id	Option to return result as generic ID	
	-gen_id <gen id>	Node generic id	
	-literal_index <literal index>	Literal index	
	-node <node id>	Node id	
	-port_type <port type>	Port type	
	-type <iport oport>	Option to specify the port as an input or output port	
<b>Description</b>	Returns the port index for the specified port type on the specified node. Returns -1 if the port is not in use or is invalid for the specified node.		
<b>Example Usage</b>	<pre>get_port_by_type -node 0 -port_type SLOAD -type iport get_port_by_type -node 0 -port_type EXTCLK -literal_index 2 -type oport</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to find port type: <string>. Specify a different port type.
	TCL_ERROR	1	ERROR: Conflicting arguments. Consult help for the Tcl command for details.
	TCL_ERROR	1	ERROR: Illegal node generic ID: %u. Specify a legal node generic ID.
	TCL_ERROR	1	ERROR: Illegal node ID: %u. Specify a legal node ID.
	TCL_ERROR	1	ERROR: Illegal port type: <string>. Specify a legal port type.
	TCL_ERROR	1	ERROR: The node you specified is a legalization node. Modification of legalization nodes is not supported.
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the read_netlist command.

### 3.1.3.10. `get_sp_pin_list` (::quartus::chip\_planner)

The following table displays information for the `get_sp_pin_list` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>get_sp_pin_list [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns a list of the pins available for use as signal probe output pins.;		
<b>Example Usage</b>	<code>get_sp_pin_list</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.3.11. `get_tile_power_setting` (::quartus::chip\_planner)

The following table displays information for the `get_tile_power_setting` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>get_tile_power_setting [-h   -help] [-long_help] [-X &lt;X location&gt;] [-Y &lt;Y location&gt;] [-gen_id &lt;gen id&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-X &lt;X location&gt;</code>	X location	
	<code>-Y &lt;Y location&gt;</code>	Y location	
	<code>-gen_id &lt;gen id&gt;</code>	Generic id	
<b>Description</b>	Returns the High-Speed/Low Power setting of the tile at the specified location.		
<b>Example Usage</b>	<code>get_tile_power_setting -gen_id 12345 get_tile_power_setting -X 12 -Y 5</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Conflicting arguments. Consult help for the Tcl command for details.
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the <code>read_netlist</code> command.

### 3.1.3.12. `read_netlist` (::quartus::chip\_planner)

The following table displays information for the `read_netlist` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>read_netlist [-h   -help] [-long_help]</code>		

*continued...*

<b>Arguments</b>	<code>-h   -help</code>		Short help
	<code>-long_help</code>		Long help with examples and possible return values
<b>Description</b>	Reads the Chip Planner netlist from the last compilation. You must open a project before using this command.		
<b>Example Usage</b>	<code>read_netlist</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to create Chip Planner netlist. Current device family does not support the Chip Planner. Specify another device family and recompile the design.
	TCL_ERROR	1	ERROR: Chip Planner (::quartus::chip_planner) is not available from the Quartus Prime Tcl Console. Run the quartus_cdb executable with commands from the ::quartus::chip_planner package from a system command prompt.
	TCL_ERROR	1	ERROR: Chip Planner is unavailable with the current license. Refer to the Licensing section of the Intel website to obtain a valid Quartus Prime license file.
	TCL_ERROR	1	ERROR: Unable to find an active revision. Make sure there is an open, active revision.
	TCL_ERROR	1	ERROR: No open project. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: Before running Chip Planner, run Analysis and Synthesis (quartus_map) for read-only use and quartus_fit to enable writable ECO changes.

### 3.1.3.13. set\_batch\_mode (::quartus::chip\_planner)

The following table displays information for the `set_batch_mode` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::chip_planner on page 90		
<b>Syntax</b>	<code>set_batch_mode [-h   -help] [-long_help] &lt;on off&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>&lt;on off&gt;</code>	Option to turn batch mode on or off	
<b>Description</b>	Sets the batch mode to On or Off.		
<b>Example Usage</b>	<code>set_batch_mode on</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Unable to find Chip Planner netlist. Read the netlist by using the <code>read_netlist</code> command.

### 3.1.4. ::quartus::design

The following table displays information for the **::quartus::design** Tcl package:

<b>Tcl Package and Version</b>	::quartus::design 1.0
<b>Description</b>	This package contains the set of Tcl functions for manipulating databases including the assignments database. Using this package makes it possible to create instance assignments without modifying the Quartus Prime Settings File (.qsf).
<b>Availability</b>	<p>This package is loaded by default in the following executable:</p> <pre>quartus_cdb</pre> <p>This package is available for loading in the following executables:</p> <pre>gacv qpro qpro_sh quartus quartus_fit quartus_map quartus_pow quartus_sh quartus_sta quartus_syn</pre>
<b>Tcl Commands</b>	<pre>design::commit_design (::quartus::design) on page 98 design::convert_partition (::quartus::design) on page 99 design::create_assignment (::quartus::design) on page 99 design::delete_assignments (::quartus::design) on page 100 design::disable_assignments (::quartus::design) on page 100 design::enable_assignments (::quartus::design) on page 101 design::export_design (::quartus::design) on page 102 design::export_partition (::quartus::design) on page 102 design::extract_metadata (::quartus::design) on page 103 design::get_assignment_info (::quartus::design) on page 104 design::get_assignment_names (::quartus::design) on page 104 design::get_assignments (::quartus::design) on page 105 design::get_entity_names (::quartus::design) on page 105 design::get_instances (::quartus::design) on page 106 design::import_design (::quartus::design) on page 106 design::import_partition (::quartus::design) on page 107 design::list_valid_snapshot_names (::quartus::design) on page 108 design::load_design (::quartus::design) on page 108 design::report_assignments (::quartus::design) on page 109 design::set_assignment_info (::quartus::design) on page 109</pre>

#### 3.1.4.1. design::commit\_design (::quartus::design)

The following table displays information for the **design::commit\_design** Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::design</a> on page 98		
<b>Syntax</b>	<code>design::commit_design [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Commit any changes to the databases to disk. Assignments created or modified on a design loaded as writeable are not saved to the databases unless you explicitly call this command.		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot -writeable design::delete_assignments [design::get_assignments -name location] design::commit_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.4.2. design::convert\_partition (::quartus::design)

The following table displays information for the design::convert\_partition Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::convert_partition [-h   -help] [-long_help] -infile <QDB file name> -outfile <QDB file name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-infile <QDB file name>	Input file name of the QDB archive (version-compatible format).	
	-outfile <QDB file name>	Output file name of the QDB archive (current-version-only format).	
<b>Description</b>	Convert a partition's QDB file in version-compatible ASCII format into a QDB file in BINARY format for current version of Quartus.		
<b>Example Usage</b>	<pre># The input QDB file is created by running design::export_partition # from compiled source design with Quartus of older or current version  project_open onewire_nf design::export_partition core_ptn -snapshot synthesized -file src_ip.qdb -compatible project_close  # Make sure you are using the same version of Quartus that will use to compile your design.  project_open onewire_nf design::convert_partition -infile src_ip.qdb -outfile ip.qdb project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified archive <string> does not exist.

### 3.1.4.3. design::create\_assignment (::quartus::design)

The following table displays information for the design::create\_assignment Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::create_assignment [-h   -help] [-long_help] [-from <from>] -name <name> -to <to> -value <value>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-from <from>	The source name of the assignment	
	-name <name>	The type name of the assignment	
	-to <to>	The destination name of the assignment	
	-value <value>	The value of the assignment	
<b>Description</b>	Create a new assignment in the assignment database		

*continued...*

<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot -writeable design::create_assignment -name location -to in1 -value PIN_A5 design::commit_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Both the -to or -from argument is required.
	TCL_ERROR	1	ERROR: Either the -to or -from argument is required.
	TCL_ERROR	1	ERROR: The -to argument is required.
	TCL_ERROR	1	ERROR: The value of an assignment cannot be empty.

### 3.1.4.4. design::delete\_assignments (::quartus::design)

The following table displays information for the design::delete\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::delete_assignments [-h   -help] [-long_help] <assignment>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	<assignment>		one or more assignment ids
<b>Description</b>	Delete one or more assignments from the assignment database		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot -writeable design::delete_assignments [design::get_assignments -name location] design::commit_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied assignment id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one periphery assignment ID must be supplied, but no assignments IDs were supplied.
	TCL_ERROR	1	ERROR: <string> assignment IDs were expected but <string> were supplied.

### 3.1.4.5. design::disable\_assignments (::quartus::design)

The following table displays information for the design::disable\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::disable_assignments [-h   -help] [-long_help] <assignment>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values

*continued...*

	<assignment>		one or more assignment ids
<b>Description</b>	Disables one or more assignments from the assignment database		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot -writeable design::disable_assignments [design::get_assignments -name location] design::commit_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied assignment id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one periphery assignment ID must be supplied, but no assignments IDs were supplied.
	TCL_ERROR	1	ERROR: <string> assignment IDs were expected but <string> were supplied.

### 3.1.4.6. design::enable\_assignments (::quartus::design)

The following table displays information for the design::enable\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::enable_assignments [-h   -help] [-long_help] <assignment>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	<assignment>		one or more assignment ids
<b>Description</b>	Enables one or more assignments from the assignment database that were previously disabled		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot -writeable design::enable_assignments [design::get_assignments -name location] design::commit_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied assignment id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one periphery assignment ID must be supplied, but no assignments IDs were supplied.
	TCL_ERROR	1	ERROR: <string> assignment IDs were expected but <string> were supplied.

### 3.1.4.7. design::export\_design (::quartus::design)

The following table displays information for the design::export\_design Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::export_design [-h   -help] [-long_help] -file <file> [-quartus_metadata <quartus_metadata>] -snapshot <snapshot> [-user_metadata <user_metadata>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-file <file>	The file.qdb to export to	
	-quartus_metadata <quartus_metadata>	A space-separated list of Quartus Metadata to export. Valid Quartus Metadata options include <none> project_information resource_utilization all>.	
	-snapshot <snapshot>	The snapshot you want to export. Valid snapshot <synthesized final>.	
	-user_metadata <user_metadata>	The absolute or relative path to the User Metadata configuration file.	
<b>Description</b>	Export the specified metadata and loaded databases for the open project and revision and snapshot to <file>.qdb in a version-compatable format. This command is available only in the quartus_cdb executable.		
<b>Example Usage</b>	<pre>project_open onewire_nf design::export_design -file onewire.qdb -snapshot synthesized project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.4.8. design::export\_partition (::quartus::design)

The following table displays information for the design::export\_partition Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::export_partition [-h   -help] [-long_help] [-exclude_pr_subblocks] -file <QDB file name> [-include_sdc_entity_in_partition] [-preserve_sdc] [-quartus_metadata <quartus_metadata>] -snapshot <Snapshot(s) to be exported> [-user_metadata <user_metadata>] <Partition to be exported>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-exclude_pr_subblocks	Exclude PR subpartitions	
	-file <QDB file name>	File name of the QDB archive.	
	-include_sdc_entity_in_partition	Preserve SDC/TCL Entity files	
	-preserve_sdc	Deprecated option to Preserve SDC/TCL Entity files	

*continued...*

	-quartus_metadata <quartus_metadata>	A space-separated list of Quartus Metadata to export. Valid Quartus Metadata options include <none project_information resource_utilization all>.	
	-snapshot <Snapshot(s) to be exported>	Snapshot(s) to be exported. Valid snapshot options include <synthesized final>.	
	-user_metadata <user_metadata>	The absolute or relative path to the User Metadata configuration file.	
	<Partition to be exported>	Name of the partition to be exported.	
<b>Description</b>	Export the metadata and IP of the specified partition and snapshot.		
<b>Example Usage</b>	<pre>project_open onewire_nf design::export_partition core_ptn -snapshot synthesized -file ip.qdb project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.4.9. design::extract\_metadata (::quartus::design)

The following table displays information for the design::extract\_metadata Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98				
<b>Syntax</b>	design::extract_metadata [-h   -help] [-long_help] -dir <dir> -file <file> [-overwrite] -type <type>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-dir <dir>	The extraction directory. This directory must exist prior to invoking this command.			
	-file <file>	The Partition Database File (.qdb) holding the metadata to be extracted.			
	-overwrite	Attempt to overwrite any existing files in the extraction directory.			
	-type <type>	The type of QDB metadata to be extracted. Legal QDB metadata types include <quartus user all>.			
<b>Description</b>	Extracts the specified metadata from the given Partition Database File (.qdb) file to the provided extraction directory.				
<b>Example Usage</b>	<pre># Create a Partition Database File (.qdb) with Quartus Metadata using the "synthesized" snapshot. project_open onewire_nf design::export_design -file onewire.qdb -snapshot synthesized -quartus_metadata all project_close  # Create the extraction directory. file mkdir "extract/dir"  # Extract all the Quartus Metadata from "onewire.qdb" to the extraction directory located at "extract/dir". design::extract_metadata -file onewire.qdb -type quartus -dir "extract/dir"</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.4.10. design::get\_assignment\_info (::quartus::design)

The following table displays information for the design::get\_assignment\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::get_assignment_info [-h   -help] [-long_help] [-from] [-name] [-to] [-value] <assignment>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-from	Return the source name of the assignment id	
	-name	Return the type name of the assignment id	
	-to	Return the destination name of the assignment id	
	-value	Return the value of the assignment id	
	<assignment>	assignment id	
<b>Description</b>	Get information about a given assignent ID		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot foreach asgn_id [design::get_assignments] {     puts "Found assignment [design::get_assignment_info -name \$asgn_id] [design::get_assignment_info -to \$asgn_id] = [design::get_assignment_info -value \$asgn_id]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied assignment id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one periphery assignment ID must be supplied, but no assignments IDs were supplied.
	TCL_ERROR	1	ERROR: <string> assignment IDs were expected but <string> were supplied.

### 3.1.4.11. design::get\_assignment\_names (::quartus::design)

The following table displays information for the design::get\_assignment\_names Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::get_assignment_names [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Get a list of valid assignment type names		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot puts "Valid assignment type names:"</pre>		

*continued...*

	<pre>foreach asgn_type [lsort [design::get_assignment_names]] {     puts \$asgn_type }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.4.12. design::get\_assignments (::quartus::design)

The following table displays information for the design::get\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::get_assignments [-h   -help] [-long_help] [-deleted] [-disabled] [-enabled] [-ignored] [-name <name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-deleted	Return only deleted assignments	
	-disabled	Return only disabled assignments	
	-enabled	Return only enabled assignments	
	-ignored	Return only ignored assignments	
	-name <name>	Return only assignments of the provided type name	
<b>Description</b>	Get a list of assignment IDs for the currently loaded design.		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot foreach asgn_id [design::get_assignments] {     puts "Found assignment [design::get_assignment_info -name \$asgn_id] [design::get_assignment_info -to \$asgn_id] = [design::get_assignment_info -value \$asgn_id]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The assignment with id <string> is not enabled.
	TCL_ERROR	1	ERROR: The supplied assignment type name <string> is invalid.
	TCL_ERROR	1	ERROR: At least one assignment type name must be supplied, but no type names were supplied.
	TCL_ERROR	1	ERROR: <string> assignment type names were expected but <string> were supplied.

### 3.1.4.13. design::get\_entity\_names (::quartus::design)

The following table displays information for the design::get\_entity\_names Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::get_entity_names [-h   -help] [-long_help] [<filter>]		

*continued...*

<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	<filter>	Object filter
<b>Description</b>	Get a list of entity names in the loaded design	
<b>Example Usage</b>	This command currently contains no example usage.	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	<b>String Return</b>	
	INFO: Operation successful	

### 3.1.4.14. design::get\_instances (::quartus::design)

The following table displays information for the `design::get_instances` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::design</a> on page 98	
<b>Syntax</b>	<code>design::get_instances [-h   -help] [-long_help] [-entity &lt;entity&gt;] [ &lt;filter&gt; ]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-entity <entity>	Return only instance names that have the supplied entity name
	<filter>	Object filter
<b>Description</b>	Get a list of instances in the loaded design	
<b>Example Usage</b>	This command currently contains no example usage.	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	<b>String Return</b>	
	INFO: Operation successful	

### 3.1.4.15. design::import\_design (::quartus::design)

The following table displays information for the `design::import_design` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::design</a> on page 98	
<b>Syntax</b>	<code>design::import_design [-h   -help] [-long_help] -file &lt;file&gt; [-overwrite] [-timing_analysis_mode]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-file <file>	The file.qdb to import from
	-overwrite	overwrites the databases in the active qdb directory
	-timing_analysis_mode	Import the design for Timing Analysis. User will not be able to generate programming file after importing design with this option. See <code>-timing_analysis_mode</code> option description below.

*continued...*

<b>Description</b>	Import all the databases from the specified <file>.qdb. If overwrite is specified then databases will be overwritten in the active qdb directory. The database revision in the <file>.qdb must match the active revision. This command is available only in the quartus_cdb executable.												
<b>Example Usage</b>	<pre># For the pro/quartus/sys/dsgn tests we need to export # a design first so there's a design to import project_open onewire_nf design::export_design -file onewire.qdb -snapshot synthesized project_close  project_open onewire_nf design::import_design -file onewire.qdb -overwrite project_close</pre>												
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: The specified archive &lt;string&gt; does not exist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Databases already exist for the specified revision &lt;string&gt;. Use -overwrite to overwrite them.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: The specified archive <string> does not exist.	TCL_ERROR	1	ERROR: Databases already exist for the specified revision <string>. Use -overwrite to overwrite them.
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>											
TCL_OK	0	INFO: Operation successful											
TCL_ERROR	1	ERROR: The specified archive <string> does not exist.											
TCL_ERROR	1	ERROR: Databases already exist for the specified revision <string>. Use -overwrite to overwrite them.											

### 3.1.4.16. design::import\_partition (::quartus::design)

The following table displays information for the design::import\_partition Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98	
<b>Syntax</b>	design::import_partition [-h   -help] [-long_help] -file <QDB file name> [-no_overwrite] <Partition to be imported>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-file <QDB file name>	File name of the QDB archive.
	-no_overwrite	Don't delete existing snapshots when importing partition
	<Partition to be imported>	Partition name at which the imported IP will be rooted.
<b>Description</b>	Import a partition into the current design.	
<b>Example Usage</b>	<pre># You need to run design::export_partition from a source design # so that there's a partition to import from  project_open onewire_nf design::export_partition root_partition -snapshot synthesized -file ip.qdb project_close  # The imported DB file can be used as root_partition  project_open onewire_nf design::import_partition root_partition -file ip.qdb project_close  # Or non-root_partition  project_open onewire_nf design::import_partition ip_sub -file ip.qdb project_close</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified archive <string> does not exist.

### 3.1.4.17. design::list\_valid\_snapshot\_names (::quartus::design)

The following table displays information for the design::list\_valid\_snapshot\_names Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::list_valid_snapshot_names [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns the list of the valid design snapshot names.		
<b>Example Usage</b>	puts "Valid design snapshot names: [design::list_valid_snapshot_names]"		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.4.18. design::load\_design (::quartus::design)

The following table displays information for the design::load\_design Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::load_design [-h   -help] [-long_help] [-flat_only] [-latest_snapshot] [-snapshot <snapshot>] [-writeable]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-flat_only	Indicates that databases should be loaded only from the flat partition.	
	-latest_snapshot	Load the latest snapshot for the design	
	-snapshot <snapshot>	Snapshot name to load database(s) from	
	-writeable	Loads databases in a writeable mode.	
<b>Description</b>	Load the databases for the currently opened project. The databases are by default loaded in read-only mode and must be loaded with the -writeable option if manipulation to the databases is desired.		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot foreach asgn_id [design::get_assignments] {     puts "Found assignment [design::get_assignment_info -name \$asgn_id] [design::get_assignment_info -to \$asgn_id] = [design::get_assignment_info -value \$asgn_id]" }</pre>		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: The specified snapshot name, <string>, is invalid.
TCL_ERROR	1	ERROR: At least one snapshot must be supplied, but no snapshots were supplied.
TCL_ERROR	1	ERROR: <string> snapshots were expected but <string> were supplied.

### 3.1.4.19. design::report\_assignments (::quartus::design)

The following table displays information for the design::report\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::report_assignments [-h   -help] [-long_help] [-deleted] [-disabled] [-enabled] [-ignored] [-name <name>] [-panel_name <name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-deleted	Report only deleted assignments	
	-disabled	Report only disabled assignments	
	-enabled	Return only enabled assignments	
	-ignored	Report only ignored assignments	
	-name <name>	The type name of the assignments to report	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	Create a report of all instance assignments in the loaded design		
<b>Example Usage</b>	<pre>project_open onewire_nf design::load_design -latest_snapshot design::report_assignments -name location</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.4.20. design::set\_assignment\_info (::quartus::design)

The following table displays information for the design::set\_assignment\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::design on page 98		
<b>Syntax</b>	design::set_assignment_info [-h   -help] [-long_help] [-disable] [-enable] [-value <value>] <assignment>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-disable	Disable the assignment	
	-enable	Enable the assignment	
	-value <value>	Set the assignment value	

*continued...*

	<assignment>		assignment id
<b>Description</b>	Set information for a given assignment ID		
<b>Example Usage</b>	<pre>project_open onewire_nf  design::load_design -latest_snapshot set asgn_id [lindex [design::get_assignments -name location] 0]   puts "Setting location of [design::get_assignment_info -to \$asgn_id] to PIN_A5"   design::set_assignment_info -value PIN_A5 \$asgn_id   puts "New location of [design::get_assignment_info -to \$asgn_id] is [design::get_assignment_info -value \$asgn_id]"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied assignment id <string> is invalid.
	TCL_ERROR	1	ERROR: The supplied assignment value <string> is invalid.
	TCL_ERROR	1	ERROR: At least one periphery assignment ID must be supplied, but no assignments IDs were supplied.
	TCL_ERROR	1	ERROR: Either the -to or -from argument is required.
	TCL_ERROR	1	ERROR: The value of an assignment cannot be empty.
	TCL_ERROR	1	ERROR: <string> assignment IDs were expected but <string> were supplied.

### 3.1.5. ::quartus::device

The following table displays information for the **::quartus::device** Tcl package:

<b>Tcl Package and Version</b>	::quartus::device 1.0
<b>Description</b>	This package contains the set of Tcl functions for accessing information from the Quartus Prime device database.
<b>Availability</b>	<p>This package is loaded by default in the following executables:</p> <pre>qpro_sh quartus_cdb quartus_edt quartus_fit quartus_ipgenerate quartus_sh quartus_sim quartus_sta</pre> <p>This package is available for loading in the following executables:</p> <pre>qpro quartus quartus_si</pre>
<b>Tcl Commands</b>	<a href="#">get_family_list</a> ( <b>::quartus::device</b> ) on page 111 <a href="#">get_part_info</a> ( <b>::quartus::device</b> ) on page 111 <a href="#">get_part_list</a> ( <b>::quartus::device</b> ) on page 112 <a href="#">report_device_info</a> ( <b>::quartus::device</b> ) on page 113 <a href="#">report_family_info</a> ( <b>::quartus::device</b> ) on page 113 <a href="#">report_part_info</a> ( <b>::quartus::device</b> ) on page 114

### 3.1.5.1. get\_family\_list (::quartus::device)

The following table displays information for the get\_family\_list Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::device on page 110				
<b>Syntax</b>	get_family_list [-h   -help] [-long_help]				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
<b>Description</b>	Returns a list of available families.				
<b>Example Usage</b>	get_family_list				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Illegal or missing <string> value, '<string>'. Specify a legal value.		

### 3.1.5.2. get\_part\_info (::quartus::device)

The following table displays information for the get\_part\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::device on page 110		
<b>Syntax</b>	get_part_info [-h   -help] [-long_help] [-default_voltage] [-device] [-family] [-family_variant] [-fast_grade_revision] [-grade_revision] [-hssi_speed_grade] [-iobank_revision] [-package] [-pdn_model_status] [-pin_count] [-pof_id] [-power_model] [-power_model_status] [-rohs_grade] [-sip_tile] [-speed_grade] [-subdevice_id_code] [-subdevice_id_mask] [-temperature_grade] <part>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-default_voltage	Option to get the default core voltage (such as 0.9V or 1.1V)	
	-device	Option to get device name (such as EP1S25 or EP1S80)	
	-family	Option to get family name (such as Stratix or Cyclone)	
	-family_variant	Option to get family variant (such as Base, E or GX)	
	-fast_grade_revision	Option to get the fast_grade_revision	
	-grade_revision	Option to get the grade_revision	
	-hssi_speed_grade	Option to get the hssi speed grade	
	-iobank_revision	Option to get the iobank_revision	
	-package	Option to get package name (such as FBGA or BGA)	
	-pdn_model_status	Option to get the power distribution network (PDN) model status (such as ADVANCE, PRELIMINARY, or FINAL)	

*continued...*

	-power_model	Option to get the power model (such as STANDARD or LOW)	
	-power_model_status	Option to get the power model status (such as PRELIMINARY or FINAL)	
	-rohs_grade	Option to get the ROHS grade (ROHS5, ROHS6, Leaded)	
	-sip_tile	Option to get SiP tile (such as L-tile, H-tile, E-tile)	
	-speed_grade	Option to get speed grade (such as 5, 6, or 7)	
	-subdevice_id_code	Option to get the subdevice_id_code	
	-subdevice_id_mask	Option to get the subdevice_id_mask	
	-temperature_grade	Option to get temperature grade of the package (such as COMMERCIAL or INDUSTRIAL)	
	<part>	Part name	
<b>Description</b>	Returns part characteristics for the specified part. If you use multiple options, the command returns a list in the following order: <family> <device> <package> <pin_count> <speed grade> <temperature_grade> <family_variant> <power_model_status> <hssi_speed_grade> <power_model> <rohs_grade>		
<b>Example Usage</b>	<pre>tcl&gt; get_part_info -family EP1S25F780C5 tcl&gt; Stratix  tcl&gt; get_part_info -family -device EP1S25F780C5 tcl&gt; Stratix EP1S25  tcl&gt; get_part_info -device -package -pin_count -speed_grade EP1S25F780C5 tcl&gt; EP1S25 FBGA 780 5</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal die location name: <string>. Specify a legal die location name.
	TCL_ERROR	1	ERROR: Illegal resource name: <string>. Specify a legal resource name.

### 3.1.5.3. `get_part_list (::quartus::device)`

The following table displays information for the `get_part_list` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::device</a> on page 110	
<b>Syntax</b>	<code>get_part_list [-h   -help] [-long_help] [-device &lt;value&gt;] [-family &lt;value&gt;] [-package &lt;value&gt;] [-pin_count &lt;value&gt;] [-speed_grade &lt;value&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-device <value>	Option to match device name
	-family <value>	Option to match family name
	-package <value>	Option to match package name
	-pin_count <value>	Option to match pin count
	-speed_grade <value>	Option to match speed grade

*continued...*

<b>Description</b>	Returns a list of available parts based on the options that are specified. Examples are as follows: Return a list of all supported parts get_part_list Return a list of all supported parts for Cyclone get_part_list -family Cyclone Return a list of all supported parts with the FBGA package and 780 pins get_part_list -package fbga -pin_count 780		
<b>Example Usage</b>	get_part_list -family "Stratix IV"		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal or missing <string> value, '<string>'. Specify a legal value.

### 3.1.5.4. report\_device\_info (::quartus::device)

The following table displays information for the `report_device_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::device on page 110		
<b>Syntax</b>	<code>report_device_info [-h   -help] [-long_help] &lt;device&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<device>	Device name	
<b>Description</b>	Returns a string value containing the report with information about the specified device, such as the following: Available parts Some additional information specific to the device		
<b>Example Usage</b>	<pre>set report [report_device_info APEX20K1000E] puts \$report</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal device name: <string>. Specify a legal device name.

### 3.1.5.5. report\_family\_info (::quartus::device)

The following table displays information for the `report_family_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::device on page 110		
<b>Syntax</b>	<code>report_family_info [-h   -help] [-long_help] &lt;family&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<family>	Family name	
<b>Description</b>	Returns a string value containing the report with information about the specified family, such as the following: Available devices Available packages Available speed grades Available pin counts Some additional information specific to the family		
<b>Example Usage</b>	<pre>set report [report_family_info Stratix] puts \$report</pre>		

*continued...*

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal family name: <string>. Specify a legal family name.

### 3.1.5.6. report\_part\_info (::quartus::device)

The following table displays information for the `report_part_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::device</a> on page 110		
<b>Syntax</b>	<code>report_part_info [-h   -help] [-long_help] &lt;part&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<part>	Part name	
<b>Description</b>	Returns a string value containing the report with information about the specified part, such as the following: Family name Device name Package name Pin count Speed grade Any additional information		
<b>Example Usage</b>	<pre>set report [report_part_info EPM1270F256I5] puts \$report</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal part name: <string>. Specify a legal part name. Use <code>report_family_info</code> , <code>report_device_info</code> , or <code>get_part_list</code> to find available parts.

### 3.1.6. ::quartus::drc

The following table displays information for the `::quartus::drc` Tcl package:

<b>Tcl Package and Version</b>	::quartus::drc 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	<p>This package is loaded by default in the following executable:  <code>quartus_da</code></p> <p>This package is available for loading in the following executables:</p> <pre>qacv qpro quartus quartus_cdb quartus_fit quartus_sta quartus_syn quartus_tlg</pre>
<b>Tcl Commands</b>	<pre>drc::add_check_op (::quartus::drc) on page 115 drc::add_check_parameter (::quartus::drc) on page 116 drc::add_object (::quartus::drc) on page 116 drc::add_object_with_properties (::quartus::drc) on page 117 drc::add_property (::quartus::drc) on page 118 drc::add_rule (::quartus::drc) on page 118 drc::add_ruleViolation (::quartus::drc) on page 119 drc::add_violation_record (::quartus::drc) on page 120 #unique_664 on page 0 drc::check_design (::quartus::drc) on page 120 drc::delete_waivers (::quartus::drc) on page 120 drc::get_objects (::quartus::drc) on page 121</pre>

*continued...*

<pre>drc::get_option (::quartus::drc) on page 122 drc::get_property (::quartus::drc) on page 122 drc::get_stage_info (::quartus::drc) on page 122 drc::get_waivers (::quartus::drc) on page 123 drc::list_properties (::quartus::drc) on page 124 drc::report_waivers (::quartus::drc) on page 124 drc::set_option (::quartus::drc) on page 125 drc::set_property (::quartus::drc) on page 125 drc::should_run_drc (::quartus::drc) on page 126 drc::update_check_op (::quartus::drc) on page 126 drc::update_rule (::quartus::drc) on page 127</pre>
---

### 3.1.6.1. drc::add\_check\_op (::quartus::drc)

The following table displays information for the `drc::add_check_op` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::drc on page 114		
<b>Syntax</b>	<code>drc::add_check_op [-h   -help] [-long_help] -bind_to_tcl_execute &lt;per_rule  per_operation&gt; [-device_families &lt;device_families&gt;] -exec_proc_name &lt;exec_proc_name&gt; [-executables &lt;executables&gt;] [-finalize_proc_name &lt;finalize_proc_name&gt;] -name &lt;name&gt; [-setup_proc_name &lt;setup_proc_name&gt;] [-stages &lt;stages&gt;] [-tcl_proc_source &lt;tcl_proc_source&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-bind_to_tcl_execute <per_rule  per_operation>	Check operation to rule binding mode	
	-device_families <device_families>	Device family, check operation is valid for	
	-exec_proc_name <exec_proc_name>	Check Operation execution proc name	
	-executables <executables>	Allowed executables, check operation could be invoked from	
	-finalize_proc_name <finalize_proc_name>	Check Operation cleanup proc name	
	-name <name>	Check Operation name	
	-setup_proc_name <setup_proc_name>	Check Operation initialization proc name	
	-stages <stages>	Allowed stages, check operation could be invoked from	
<b>Description</b>	Add Check Operation .		
	<code>add_check_op -name {TEST} -bind_to_tcl_execute {per_operation} -exec_proc_name {TEST_EXEC_PROC}</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.2. drc::add\_check\_parameter (::quartus::drc)

The following table displays information for the `drc::add_check_parameter` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::add_check_parameter [-h   -help] [-long_help] -check_op &lt;check_op&gt; [-is_user &lt;is_user&gt;] [-param_description &lt;param_description&gt;] -param_name &lt;param_name&gt; [-param_range &lt;param_range&gt;] -param_value &lt;param_value&gt; -value_type &lt;integer double string string_list bool&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-check_op <check_op>	Check Operation name	
	-is_user <is_user>	Is parameter user visible	
	-param_description <param_description>	Parameter description	
	-param_name <param_name>	Parameter name	
	-param_range <param_range>	Legal range for parameter	
	-param_value <param_value>	Default value for parameter	
	-value_type <integer double string string_list bool>	Parameter value type integer double string string_list bool	
<b>Description</b>	Add Check Operation .		
<b>Example Usage</b>	<pre>add_check_parameter -check_op {TEST} -param_name {TEST_PARAM} -value_type {string} -param_value {TEST_VALUE}</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.3. drc::add\_object (::quartus::drc)

The following table displays information for the `drc::add_object` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::add_object [-h   -help] [-long_help] [-bind_to_tcl_execute &lt;bind_to_tcl_execute&gt;] [-category &lt;category&gt;] [-name &lt;name&gt;] [-number &lt;number&gt;] [-parent &lt;parent&gt;] -type &lt;type&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-bind_to_tcl_execute <bind_to_tcl_execute>	Specifies the Tcl execution mode of the new check_operation DRC object (none / per_rule / per_operation).	
	-category <category>	The category of the new DRC object.	
	-name <name>	The name of the new check_operation/rule_set DRC object.	
	-number <number>	The numeric part of the new DRC object's name.	
	<i>continued...</i>		

	-parent <parent>	The parent object of the new violation_record DRC object.	
	-type <type>	The type of the new DRC object.	
<b>Description</b>	Adds a new DRC object.		
<b>Example Usage</b>	<code>drc::add_object -type rule -category CLK -number 1</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to add DRC object.
	TCL_ERROR	1	ERROR: Invalid object type '<string>'.
	TCL_ERROR	1	ERROR: Category is needed for finding RULE or VIOLATION.
	TCL_ERROR	1	ERROR: Object name is needed.
	TCL_ERROR	1	ERROR: Object number is needed.
	TCL_ERROR	1	ERROR: Parent object is needed for violation record.

### 3.1.6.4. drc::add\_object\_with\_properties (::quartus::drc)

The following table displays information for the `drc::add_object_with_properties` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::add_object_with_properties [-h   -help] [-long_help] [-bind_to_tcl_execute &lt;bind_to_tcl_execute&gt;] [-category &lt;category&gt;] [-name &lt;name&gt;] [-number &lt;number&gt;] [-parent &lt;parent&gt;] -properties &lt;properties&gt; -type &lt;type&gt;</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-bind_to_tcl_execute <bind_to_tcl_execute>		Specifies the Tcl execution mode of the new check_operation DRC object (none / per_rule / per_operation).
	-category <category>		The category of the new DRC object.
	-name <name>		The name of the new check_operation/rule_set DRC object.
	-number <number>		The numeric part of the new DRC object's name.
	-parent <parent>		The parent object of the new violation_record DRC object.
	-properties <properties>		The property list of the new violation_record DRC object.
	-type <type>		The type of the new DRC object.
<b>Description</b>	Adds a new DRC object with properties.		
<b>Example Usage</b>	<pre>drc::add_object_with_properties -type violation_record -name &lt;violation&gt; -parent &lt;result_record_id&gt; -properties [list [list "fields" [list "HighFanout_DRV1", "1530"]], [list "fields:1:location_schema" "mod_A mod_A_1 out"]] drc::add_object_with_properties -type violation_record -name &lt;violation&gt; -parent &lt;result_record_id&gt; -properties {"fields" {"HighFanout_DRV1", "1530"}}  {"fields:1:location_schema" "mod_A mod_A_1 out"}}</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: Fail to add DRC object.
TCL_ERROR	1	ERROR: Invalid object type '<string>'.
TCL_ERROR	1	ERROR: Property list should consist of name-value pairs.
TCL_ERROR	1	ERROR: Category is needed for finding RULE or VIOLATION.
TCL_ERROR	1	ERROR: Object name is needed.
TCL_ERROR	1	ERROR: Object number is needed.
TCL_ERROR	1	ERROR: Parent object is needed for violation record.

### 3.1.6.5. drc::add\_property (::quartus::drc)

The following table displays information for the `drc::add_property` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::drc on page 114		
<b>Syntax</b>	<code>drc::add_property [-h   -help] [-long_help] -name &lt;name&gt; -object &lt;object&gt; -value &lt;value&gt; [-value_type &lt;value_type&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The DRC object to which the new property belongs.	
	-value <value>	property value.	
	-value_type <value_type>	property value type.	
<b>Description</b>	Add a property (a name/value pair) to a generic DRC object		
<b>Example Usage</b>	<code>drc::add_property -object &lt;DRC object&gt; -name &lt;property name&gt; -value &lt;property value&gt; -value_type &lt;value type&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to add object property.
	TCL_ERROR	1	ERROR: Invalid DRC object '%s'.
	TCL_ERROR	1	ERROR: Invalid property value.
	TCL_ERROR	1	ERROR: Invalid property value type.
	TCL_ERROR	1	ERROR: Property '<string>' already exists. Cannot be added.

### 3.1.6.6. drc::add\_rule (::quartus::drc)

The following table displays information for the `drc::add_rule` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::drc on page 114		
<b>Syntax</b>	<code>drc::add_rule [-h   -help] [-long_help] -category &lt;category&gt; -check_operation &lt;check_operation&gt; [-description &lt;description&gt; ] -id &lt;id&gt; [-is_default &lt;is_default&gt; ] [-recommendation &lt;recommendation&gt; ] -severity &lt;severity&gt; [-short_description &lt;short_description&gt; ]</code>		
<i>continued...</i>			

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-category &lt;category&gt;</code>	Rule category	
	<code>-check_operation &lt;check_operation&gt;</code>	Check operation associated	
	<code>-description &lt;description&gt;</code>	Detailed description	
	<code>-id &lt;id&gt;</code>	Rule ID	
	<code>-is_default &lt;is_default&gt;</code>	Is Default Rule	
	<code>-recommendation &lt;recommendation&gt;</code>	Detailed recommendation	
	<code>-severity &lt;severity&gt;</code>	Rule Severity	
	<code>-short_description &lt;short_description&gt;</code>	Rule title	
<b>Description</b>	Add Check Operation .		
<b>Example Usage</b>	<pre>add_rule -category {TEST} -id {TEST_ID} -check_operation {TEST_CHECK_OP} -severity {TEST_SEVERITY}</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.7. drc::add\_ruleViolation (::quartus::drc)

The following table displays information for the `drc::add_ruleViolation` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114				
<b>Syntax</b>	<code>drc::add_ruleViolation [-h   -help] [-long_help] [-argument_spec &lt;argument_spec&gt;] [-category &lt;category&gt;] [-format_msg &lt;format_msg&gt;] [-id &lt;id&gt;] [-num_args &lt;num_args&gt;] [-object &lt;rule_object&gt;]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-argument_spec &lt;argument_spec&gt;</code>	Argument specifications {{type name description [sort_method] [order]}...}			
	<code>-category &lt;category&gt;</code>	Rule category for violation			
	<code>-format_msg &lt;format_msg&gt;</code>	Violation message format			
	<code>-id &lt;id&gt;</code>	Rule ID for violation			
	<code>-num_args &lt;num_args&gt;</code>	Number of arguments			
	<code>-object &lt;rule_object&gt;</code>	Rule object for violation			
<b>Description</b>	Add Rule Violation.				
<b>Example Usage</b>	<pre>add_ruleViolation</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.6.8. drc::addViolationRecord (::quartus::drc)

The following table displays information for the `drc::addViolationRecord` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::addViolationRecord [-h   -help] [-long_help] -parent &lt;parent&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-parent <parent>	Parent result record for violation	
<b>Description</b>	Add Violation Record.		
<b>Example Usage</b>	<code>addViolationRecord -parent {PARENT_RESULT_RECORD_ID}</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.9. drc::checkDesign (::quartus::drc)

The following table displays information for the `drc::checkDesign` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::checkDesign [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Run the specified rule set on a snapshot for the open project.		
<b>Example Usage</b>	<code>drc::checkDesign [-rule_set my_ruleset (if missing, the rule set is default)] [-executable quartus_sta (if missing, the default is the current process name)] [-rpt_file &lt;DA DRC ASCII report file path&gt;]</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to complete checks.

### 3.1.6.10. drc::deleteWaivers (::quartus::drc)

The following table displays information for the `drc::deleteWaivers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::deleteWaivers [-h   -help] [-long_help] [-all] [-owner &lt;owner&gt;] [-query_string &lt;query_string&gt;] [-rule_id &lt;rule_id&gt;] [-stages &lt;stages&gt;] [-tag &lt;tag&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<i>continued...</i>			

	-all	The flag needed explicitly for deleting all the waivers without other arguments	
	-owner <owner>	User ID of creator. Meant for waiver audit trail.	
	-query_string <query_string>	Query string uses one or more violation column arguments to define patterns of violations that should be ignored.	
	-rule_id <rule_id>	Alpha-numeric rule ID pattern to define the scope of this waiver.	
	-stages <stages>	one or more stage(s) where rules are defined, for which the waiver can be applied to.	
	-tag <tag>	User-specified tags for tracking different types of violations across the whole project.	
<b>Description</b>	Delete waivers based on the input arguments		
<b>Example Usage</b>	<pre># This delete all the waivers. ::drc::delete_waivers</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.11. drc::get\_objects (::quartus::drc)

The following table displays information for the drc::get\_objects Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::drc on page 114		
<b>Syntax</b>	drc::get_objects [-h   -help] [-long_help] [-category <category>] [-name <name>] [-number <number>] [-parent <parent>] -type <type>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-category <category>	The category of the rule/violation DRC objects to be retrieved.	
	-name <name>	The literal name of the check_operation/rule_set DRC objects to be retrieved.	
	-number <number>	The numerical part of the name of the rule/violation DRC objects to be retrieved.	
	-parent <parent>	The parent object of the violation_record DRC object to be retrieved.	
	-type <type>	The type of DRC objects to be retrieved.	
<b>Description</b>	Get a list of specific existing DRC objects		
<b>Example Usage</b>	drc::get_objects -type rule		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to get DRC objects.
	TCL_ERROR	1	ERROR: Parent object is needed for violation record.

### 3.1.6.12. drc::get\_option (::quartus::drc)

The following table displays information for the `drc::get_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::get_option [-h   -help] [-long_help] -name &lt;name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	option name.	
<b>Description</b>	Get option for the DRC system		
<b>Example Usage</b>	<code>drc::get_option -name &lt;option name&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: DRC option '<string>' is invalid.

### 3.1.6.13. drc::get\_property (::quartus::drc)

The following table displays information for the `drc::get_property` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::get_property [-h   -help] [-long_help] -name &lt;name&gt; -object &lt;object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The DRC object to which the property belongs.	
<b>Description</b>	Get the value of a generic DRC object's property.		
<b>Example Usage</b>	<code>drc::get_property -object &lt;DRC object&gt; -name &lt;property name&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid DRC object '%s'.
	TCL_ERROR	1	ERROR: Property '<string>' is invalid.

### 3.1.6.14. drc::get\_stage\_info (::quartus::drc)

The following table displays information for the `drc::get_stage_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::get_stage_info [-h   -help] [-long_help] [-executable &lt;executable&gt;] [-rule_set &lt;rule_set&gt;] [-snapshot &lt;snapshot&gt;] [-stage &lt;stage&gt;]</code>		
<i>continued...</i>			

<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-executable &lt;executable&gt;</code>	The executable name to find stages.
	<code>-rule_set &lt;rule_set&gt;</code>	The rule set name to find stages.
	<code>-snapshot &lt;snapshot&gt;</code>	The snapshot name to find stages.
	<code>-stage &lt;stage&gt;</code>	The stage name to get executable, snapshot, and rule set.
<b>Description</b>	The Utility to find stage info in terms of all available stages, and executables and snapshots associate with snapshots. No error info provided.	
<b>Example Usage</b>	<pre># Get all the stage names ::drc::get_stage_info  # Get executable and snapshot for a stage ::drc::get_stage_info -stage \$stage_name  # Get stages with the input executable name ::drc::get_stage_info -executable \$executable_name  # Get stages with the input snapshot name ::drc::get_stage_info -snapshot \$snapshot_name  # Get stage with the input snapshot and executable name ::drc::get_stage_info -snapshot \$snapshot_name -executable \$executable_name</pre>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	<b>String Return</b>	
	INFO: Operation successful	

### 3.1.6.15. `drc::get_waivers` (::quartus::drc)

The following table displays information for the `drc::get_waivers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::drc on page 114	
<b>Syntax</b>	<code>drc::get_waivers [-h   -help] [-long_help] [-owner &lt;owner&gt;] [-query_string &lt;query_string&gt;] [-rule_id &lt;rule_id&gt;] [-stages &lt;stages&gt;] [-tag &lt;tag&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-owner &lt;owner&gt;</code>	User ID of creator. Meant for waiver audit trail.
	<code>-query_string &lt;query_string&gt;</code>	Query string uses one or more violation column arguments to define patterns of violations that should be ignored.
	<code>-rule_id &lt;rule_id&gt;</code>	Alpha-numeric rule ID pattern to define the scope of this waiver.
	<code>-stages &lt;stages&gt;</code>	one or more stage(s) where rules are defined, for which the waiver can be applied to.
	<code>-tag &lt;tag&gt;</code>	User-specified tags for tracking different types of violations across the whole project.
<b>Description</b>	List all the waiver objects found in the memory filtered based on the input arguments.	
<b>Example Usage</b>	<pre>set waiver_objs [::drc::get_waivers]</pre>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	<b>String Return</b>	
	INFO: Operation successful	

### 3.1.6.16. drc::list\_properties (::quartus::drc)

The following table displays information for the `drc::list_properties` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::list_properties [-h   -help] [-long_help] -object &lt;object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-object <object>	The DRC object whose properties are returned as a list of property names.	
<b>Description</b>	Get the list of property names from a generic DRC object		
<b>Example Usage</b>	<code>drc::list_properties -object &lt;DRC object&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid DRC object '%'. %

### 3.1.6.17. drc::report\_waivers (::quartus::drc)

The following table displays information for the `drc::report_waivers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::report_waivers [-h   -help] [-long_help] -file &lt;file&gt; [-force] [-owner &lt;owner&gt;] [-query_string &lt;query_string&gt;] [-rule_id &lt;rule_id&gt;] [-stages &lt;stages&gt;] [-tag &lt;tag&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-file <file>	The file where to store all the waivers found in the memory	
	-force	the flag which can force to update DESIGN_ASSISTANT_WAIVER_FILE	
	-owner <owner>	User ID of creator. Meant for waiver audit trail.	
	-query_string <query_string>	Query string uses one or more violation column arguments to define patterns of violations that should be ignored.	
	-rule_id <rule_id>	Alpha-numeric rule ID pattern to define the scope of this waiver.	
	-stages <stages>	one or more stage(s) where rules are defined, for which the waiver can be applied to.	
	-tag <tag>	User-specified tags for tracking different types of violations across the whole project.	
<b>Description</b>	Output the definition of all the waivers in the memory to a file filtered based on the input arguments.		

*continued...*

<b>Example Usage</b>	<code>::drc::report_waivers -file waivers.dawf</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.18. drc::set\_option (::quartus::drc)

The following table displays information for the `drc::set_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::set_option [-h   -help] [-long_help] -name &lt;name&gt; -value &lt;value&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	option name.	
	-value <value>	option value.	
<b>Description</b>	Set options for the DRC system		
<b>Example Usage</b>	<code>drc::set_option -name &lt;option name&gt; -value &lt;option value&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: DRC option '<string>' is invalid.

### 3.1.6.19. drc::set\_property (::quartus::drc)

The following table displays information for the `drc::set_property` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::set_property [-h   -help] [-long_help] -name &lt;name&gt; -object &lt;object&gt; -value &lt;value&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The DRC object to which the property belongs.	
	-value <value>	property value.	
<b>Description</b>	Update property in term of name/value pair to a generic DRC object		
<b>Example Usage</b>	<code>drc::set_property -object &lt;DRC object&gt; -name &lt;property name&gt; -value &lt;property value&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid DRC object '%s'.
	TCL_ERROR	1	ERROR: Property '<string>' is invalid.

### 3.1.6.20. drc::should\_run\_drc (::quartus::drc)

The following table displays information for the `drc::should_run_drc` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::should_run_drc [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Determine if DRC should run.		
<b>Example Usage</b>	<code>should_run_drc</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.21. drc::update\_check\_op (::quartus::drc)

The following table displays information for the `drc::update_check_op` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::drc</a> on page 114		
<b>Syntax</b>	<code>drc::update_check_op [-h   -help] [-long_help] [-device_families &lt;device_families&gt;] [-executables &lt;executables&gt;] -name &lt;name&gt; [-stages &lt;stages&gt;] [-tcl_proc_source &lt;tcl_proc_source&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-device_families &lt;device_families&gt;</code>	Device family, check operation is valid for	
	<code>-executables &lt;executables&gt;</code>	Allowed executables, check operation could be invoked from	
	<code>-name &lt;name&gt;</code>	Check operation name	
	<code>-stages &lt;stages&gt;</code>	Allowed stages, check operation could be invoked from	
	<code>-tcl_proc_source &lt;tcl_proc_source&gt;</code>	Check operation proc source file name	
<b>Description</b>	Add Check Operation .		
<b>Example Usage</b>	<code>update_check_op -name {TEST}</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.6.22. drc::update\_rule (::quartus::drc)

The following table displays information for the `drc::update_rule` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::drc</code> on page 114		
<b>Syntax</b>	<code>drc::update_rule [-h   -help] [-long_help] -category &lt;category&gt; [-description &lt;description&gt;] [-exec_proc &lt;exec_proc&gt;] [-finalize_proc &lt;finalize_proc&gt;] -id &lt;id&gt; [-is_user &lt;is_user&gt;] [-param_description &lt;param_description&gt;] [-param_value &lt;param_value&gt;] [-parameter &lt;parameter&gt;] [-recommendation &lt;recommendation&gt;] [-setup_proc &lt;setup_proc&gt;] [-short_description &lt;short_description&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-category <category>	Rule category	
	-description <description>	Rule description	
	-exec_proc <exec_proc>	Execution proc for the rule	
	-finalize_proc <finalize_proc>	Cleanup proc for the rule	
	-id <id>	Rule ID	
	-is_user <is_user>	Is rule parameter user visible	
	-param_description <param_description>	Parameter description	
	-param_value <param_value>	Parameter default value	
	-parameter <parameter>	Parameter for the rule	
	-recommendation <recommendation>	Rule recommendation	
<b>Description</b>	Update Check Operation.		
	<code>update_rule -category {TEST} -id {TEST_ID}</code>		
<b>Example Usage</b>			
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7. ::quartus::eco

The following table displays information for the `::quartus::eco` Tcl package:

<b>Tcl Package and Version</b>	::quartus::eco 1.0
<b>Description</b>	This package contains commands to perform Engineering Change Orders on a Post-Fit netlist. ECO compilations are supported on Intel Stratix 10 and Intel Agilex device families.
<b>Availability</b>	This package is available for loading in the following executable: <code>quartus_fit</code>
<i>continued...</i>	

<b>Tcl Commands</b>	<pre> adjust_pll_refclk (::quartus::eco) on page 128 #unique_665 on page 0 create_wirelut (::quartus::eco) on page 128 eco_reroute (::quartus::eco) on page 129 eco_unload_design (::quartus::eco) on page 130 fitter_report_timing (::quartus::eco) on page 130 fitter_timing_summary (::quartus::eco) on page 131 get_available_snapshots (::quartus::eco) on page 131 get_eco_checkpoint (::quartus::eco) on page 132 get_loaded_snapshot (::quartus::eco) on page 132 get_lutmask_equation (::quartus::eco) on page 132 get_node_location (::quartus::eco) on page 133 make_connection (::quartus::eco) on page 133 modify_io_current_strength (::quartus::eco) on page 134 modify_io_delay_chain (::quartus::eco) on page 134 modify_io_slew_rate (::quartus::eco) on page 135 #unique_666 on page 0 place_node (::quartus::eco) on page 135 remove_connection (::quartus::eco) on page 136 remove_node (::quartus::eco) on page 136 report_connections (::quartus::eco) on page 137 report_legal_locations (::quartus::eco) on page 138 report_nodes_at_location (::quartus::eco) on page 138 report_ports (::quartus::eco) on page 139 report_routing (::quartus::eco) on page 139 report_unplaced_nodes (::quartus::eco) on page 140 restore_eco_checkpoint (::quartus::eco) on page 140 unplace_node (::quartus::eco) on page 141 update_mif_files (::quartus::eco) on page 141 </pre>
---------------------	---

### 3.1.7.1. `adjust_pll_refclk (::quartus::eco)`

The following table displays information for the `adjust_pll_refclk` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>adjust_pll_refclk [-h   -help] [-long_help] -refclk &lt;refclk_freq&gt; -to &lt;iopl_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-refclk <refclk_freq>	New refclk frequency value in MHz	
	-to <iopl_name>	Instance name of upstream IOPLL to be adjusted	
<b>Description</b>	The <code>adjust_pll_refclk</code> command can change IOPLL frequencies by modifying the input reference clock frequency. Assumptions and Limitations: - The original refclk/outclk ratios will be maintained - None of the IOPLLs being reconfigured can generate IP clocks (the frequencies of the LVDS, PhyLite and EMIF clocks cannot change) - Cascaded IOPLLs must be connected directly (no clock gates in between them) - IOPLLs cannot be in 'nondedicated' compensation modes - For all IOPLL outclks duty cycle = 50 and phase shift = 0		
<b>Example Usage</b>	<pre> load_package eco adjust_pll_refclk -to "*pll_main*" -refclk 100 </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.2. `create_wirelut (::quartus::eco)`

The following table displays information for the `create_wirelut` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>create_wirelut [-h   -help] [-long_help] -from &lt;source_output_net_name&gt; [-location &lt;location&gt;] [-name &lt;node_name&gt;] -port &lt;dest_node_port&gt; -to &lt;dest_node_name&gt;</code>		
	<i>continued...</i>		

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-from &lt;source_output_net_name&gt;</code>	Source of the connection	
	<code>-location &lt;location&gt;</code>	PLACE_REGION assignment type location	
	<code>-name &lt;node_name&gt;</code>	Name of the new node	
	<code>-port &lt;dest_node_port&gt;</code>	Input port name of the destination node	
	<code>-to &lt;dest_node_name&gt;</code>	Name of the destination atom node	
<b>Description</b>	The create_wirelut command will create and insert a wire LUT node in the specified connection. The ECO Fitter will place the newly created LUT and route the modified connections automatically, if -location argument is specified. Otherwise, the new wire LUT should be placed with place_node command. The name for the new node is hierarchy based, and the ECO Fitter will try to infer the name hierarchy. For example, if a node a b c d needs to be created, users should make sure that hierarchy a b c exists in the netlist. If the source or destination node lies under a partition, the new wire LUT will be inserted under that partition.		
<b>Example Usage</b>	<pre>create_wirelut -name my_wirelut -from src_output -to dest_node -port D -location "X136 Y63 X149 Y82" create_wirelut -name my_wirelut -from src_output -to dest_node -port D place_node -name my_wirelut</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.3. eco\_reroute (::quartus::eco)

The following table displays information for the eco\_reroute Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>eco_reroute [-h   -help] [-long_help] [-hold_slack &lt;hold_slack&gt;] [-keep_best] -node &lt;node&gt; -port &lt;port&gt; [-setup_slack &lt;setup_slack&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-hold_slack &lt;hold_slack&gt;</code>	targeted hold slack (ns)	
	<code>-keep_best</code>	keep best attempted slack if target slack not met	
	<code>-node &lt;node&gt;</code>	node name	
	<code>-port &lt;port&gt;</code>	port name	
	<code>-setup_slack &lt;setup_slack&gt;</code>	targeted setup slack (ns)	
<b>Description</b>	The eco_reroute command reroutes the item with user specified slacks.		
<b>Example Usage</b>	<pre>eco_reroute -node &lt;node_name&gt; -port &lt;port_name&gt; eco_reroute -node &lt;node_name&gt; -port &lt;port_name&gt; -setup_slack &lt;setup_slack&gt; -hold_slack &lt;hold_slack&gt; -max_iteration &lt;max_iteration&gt; -keep_best eco_reroute -node ff -port D -hold_slack 0.3 -setup_slack -2 -max_iteration 10</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.4. eco\_unload\_design (::quartus::eco)

The following table displays information for the eco\_unload\_design Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	eco_unload_design [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	The eco_unload_design command unloads the current design. It also discards all changes that were made but not committed.		
<b>Example Usage</b>	eco_unload_design		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.5. fitter\_report\_timing (::quartus::eco)

The following table displays information for the fitter\_report\_timing Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	fitter_report_timing [-h   -help] [-long_help] [-detail <summary path_only path_and_clock full_path>] [-extra_info <basic all none>] [-from <names>] [-from_clock <names>] [-hold] [-npaths <number>] [-panel_name <name>] [-recovery] [-removal] [-setup] [-through <names>] [-to <names>] [-to_clock <names>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-detail <summary path_only path_and_clock full_path>	Option to determine how much detail should be shown in the path report	
	-extra_info <basic all none>	Option to determine how much detail should be shown in the Extra Info report	
	-from <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	-from_clock <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	-hold	Option to report clock hold paths	
	-npaths <number>	Specifies the number of paths to report (default=1)	
	-panel_name <name>	Sends the results to the timing report	
	-recovery	Option to report recovery paths	
	-removal	Option to report removal paths	
	-setup	Option to report clock setup paths	
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)	

*continued...*

	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	-to_clock <names>	Valid destinations (string patterns are matched using Tcl string matching)	
<b>Description</b>	Reports the worst-case paths and associated slack.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.6. fitter\_timing\_summary (::quartus::eco)

The following table displays information for the `fitter_timing_summary` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>fitter_timing_summary [-h   -help] [-long_help] [-hold] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-setup]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-hold	Option to report clock hold paths	
	-panel_name <name>	Sends the results to the timing summary	
	-recovery	Option to report recovery paths	
	-removal	Option to report removal paths	
	-setup	Option to report clock setup paths	
<b>Description</b>	Reports the worst-case Clock Setup and Clock Hold slacks and endpoint TNS (total negative slack) per clock domain. Total negative slack is the sum of all slacks less than zero for either destination registers or ports in the clock domain.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.7. get\_available\_snapshots (::quartus::eco)

The following table displays information for the `get_available_snapshots` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>get_available_snapshots [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	The <code>get_available_snapshots</code> command reports all available snapshots.		

*continued...*

<b>Example Usage</b>	get_available_snapshots		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.8. get\_eco\_checkpoint (::quartus::eco)

The following table displays information for the `get_eco_checkpoint` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>get_eco_checkpoint [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	The <code>get_eco_checkpoint</code> command prints the current checkpoint id in the console. It also returns the checkpoint id in a tcl object.		
<b>Example Usage</b>	get_eco_checkpoint		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.9. get\_loaded\_snapshot (::quartus::eco)

The following table displays information for the `get_loaded_snapshot` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>get_loaded_snapshot [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	The <code>report_loaded_snapshot</code> command reports the loaded snapshot.		
<b>Example Usage</b>	get_loaded_snapshot		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.10. get\_lutmask\_equation (::quartus::eco)

The following table displays information for the `get_lutmask_equation` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>get_lutmask_equation [-h   -help] [-long_help] -name &lt;name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<i>continued...</i>			

	-name <name>	name of the lut
<b>Description</b>	The get_lutmask_equation command reports the LUT equation of a given LUT.	
<b>Example Usage</b>	get_lutmask_equation -name <name_string>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
		INFO: Operation successful

### 3.1.7.11. get\_node\_location (::quartus::eco)

The following table displays information for the get\_node\_location Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127	
<b>Syntax</b>	get_node_location [-h   -help] [-long_help] -name <node_name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-name <node_name>	Name of node to get location for
<b>Description</b>	The get_node_location command will print the location of the node in the console and return the location in a tcl object. If the node is not placed, then the string "Unplaced" will be printed and returned.	
<b>Example Usage</b>	get_node_location -name node	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
		INFO: Operation successful

### 3.1.7.12. make\_connection (::quartus::eco)

The following table displays information for the make\_connection Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127	
<b>Syntax</b>	make_connection [-h   -help] [-long_help] [-from <output_net_name>] -port <dest_node_port> [-tieoff <VCC GND>] -to <dest_node_name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-from <output_net_name>	Source of the connection
	-port <dest_node_port>	Input port name of the destination node
	-tieoff <VCC GND>	VCC or GND tieoff
	-to <dest_node_name>	Name of the destination node
<b>Description</b>	The make_connection command will connect the source signal to the destination block port. If the port has an existing connection, the command will remove the previous connection and connect it to the specified signal. make_connection expects 3 arguments: from - output net of the source block of the new connection, OR tieoff - tieoff value to - name of the destination block port - the input port name of the destination block Note that the changed path will be routed immediately. If the path contains a node that is created during ECO compilation, then the paths will be routed after the node is placed with place_node command.	

*continued...*

<b>Example Usage</b>	<pre>make_connection -from top a_out -to top x -port D This example will connect top a_out to the D input port of node top x.  make_connection -tieoff VCC -to top x -port D This example will tie the D port of node top x to VCC, either internally or via lcell.</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.13. `modify_io_current_strength` (::quartus::eco)

The following table displays information for the `modify_io_current_strength` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>modify_io_current_strength [-h   -help] [-long_help] -to &lt;dest_pin_name&gt; &lt;current_strength&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-to <dest_pin_name>	Name of the destination pin node	
	<current_strength>	Current strength value	
<b>Description</b>	The <code>modify_io_current_strength</code> command will modify the current strength of the targeted pin. <code>modify_io_current_strength</code> expects 1 positional argument and an option argument: <code>to</code> - the name of the destination pin		
<b>Example Usage</b>	<code>modify_io_current_strength 3mA -to top ipin</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.14. `modify_io_delay_chain` (::quartus::eco)

The following table displays information for the `modify_io_delay_chain` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>modify_io_delay_chain [-h   -help] [-long_help] -to &lt;dest_pin_name&gt; -type &lt;input output oe io_12_lane_input_data io_12_lane_input_strobe&gt; &lt;delay_chain&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-to <dest_pin_name>	Name of the destination pin node	
	-type <input output oe  io_12_lane_input_data  io_12_lane_input_strobe>	Type of the pin	
	<delay_chain>	Delay chain value	

*continued...*

<b>Description</b>	The modify_io_delay_chain command will modify the delay chain setting of the targeted pin with the specified type. modify_io_delay_chain expects 1 positional argument and 2 option arguments: to - the name of the destination pin type - the type of the pin		
<b>Example Usage</b>	modify_io_delay_chain 3 -to top ipin -type INPUT		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.15. modify\_io\_slew\_rate (::quartus::eco)

The following table displays information for the `modify_io_slew_rate` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>modify_io_slew_rate [-h   -help] [-long_help] -to &lt;dest_pin_name&gt; &lt;slew_rate&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-to <dest_pin_name>	Name of the destination pin node	
	<slew_rate>	Slew rate value	
<b>Description</b>	The <code>modify_io_slew_rate</code> command will modify the slew rate of the targeted pin. <code>modify_io_slew_rate</code> expects 1 positional argument and an option argument: to - the name of the destination pin		
<b>Example Usage</b>	modify_io_slew_rate 1 -to top ipin		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.16. place\_node (::quartus::eco)

The following table displays information for the `place_node` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>place_node [-h   -help] [-long_help] [-force] [-location &lt;location&gt;] -name &lt;node_name&gt; [-sample &lt;sample&gt;] [-timing_driven]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-force	Force overwriting existing location constraint on the node	
	-location <location>	Exact location or region	
	-name <node_name>	Name of the new node	
	-sample <sample>	Number of locations to try to place the node	
	-timing_driven	Try to place the node at a location meeting timing	
<b>Description</b>	The <code>place_node</code> command will place the specified node either automatically, or within the region if - location is specified. If -sample is specified, then the command will randomly pick locations and place the node at the first valid location found. If -timing_driven is specified then the command will try to place the node at a location that meets timing. If none of the locations meet timing, then the node		

*continued...*

	will be placed at the location with the highest slacks. The place_node command can be used on nodes that have been placed. -location argument takes in an exact location (-location "X20 Y20"), a region (-location "X20 Y20 X30 Y30"), or an ALM sublocation (-location "FF_X20_Y20_N10"). -force argument will force overwrite existing location constraints on the node, if any. -force will not overwrite Partial Reconfiguration regions.						
<b>Example Usage</b>	<pre>place_node -name eco_new_lut -location "X136 Y63 X149 Y82" place_node -name eco_new_lut -location "X5 Y10" place_node -name eco_new_lut -location "X5 Y10" -timing_driven place_node -name eco_new_ff -location "FF_X20_Y20_N10" place_node -name eco_new_ff -location "FF_X20_Y20_N10" -force place_node -name eco_new_lut -location "X5 Y10 X100 Y200" -sample 10 place_node -name eco_new_ff -location "X5 Y10 X100 Y200" -sample 100 -timing_driven</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.7.17. remove\_connection (::quartus::eco)

The following table displays information for the remove\_connection Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127								
<b>Syntax</b>	<code>remove_connection [-h   -help] [-long_help] -from &lt;output_net_name&gt; -port &lt;dest_node_port&gt; -to &lt;dest_node_name&gt;</code>								
<b>Arguments</b>	-h   -help	Short help							
	-long_help	Long help with examples and possible return values							
	-from <output_net_name>	Source of the connection							
	-port <dest_node_port>	Input port name of the destination node							
	-to <dest_node_name>	Name of the destination atom node							
<b>Description</b>	The remove_connection command will disconnect the source signal from the destination block port, and set the input port to a disconnected state. remove_connection expects 3 arguments: from - output net of the source block of the new connection to - name of the destination block port - the input port name of the destination block Note: Connection path containing Hyper Registers is not allowed to be removed.								
<b>Example Usage</b>	<pre>remove_connection -from top a_out -to top x -port D</pre> <p>This example will disconnect top a_out from the D input port of node top x, and set top x:D to a disconnected state.</p>								
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful		
Code Name	Code	String Return							
TCL_OK	0	INFO: Operation successful							

### 3.1.7.18. remove\_node (::quartus::eco)

The following table displays information for the remove\_node Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127		
<b>Syntax</b>	<code>remove_node [-h   -help] [-long_help] -name &lt;node_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <node_name>	Name of the node to remove	

*continued...*

<b>Description</b>	The remove_node command will remove the specified node from final netlist. The node's source and destination signals will be disconnected.		
<b>Example Usage</b>	remove_node -name my_ff		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.19. report\_connections (::quartus::eco)

The following table displays information for the report\_connections Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	report_connections [-h   -help] [-long_help] [-from <from>] [-from_port <from_port>] [-limit <limit>] [-return_result] [-timing] [-to <to>] [-to_port <to_port>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-from <from>	Name of the source node	
	-from_port <from_port>	Name of the source port	
	-limit <limit>	Limit number of output connections reported	
	-return_result	Return the result in a tcl object	
	-timing	Report slacks	
	-to <to>	Name of the destination node	
	-to_port <to_port>	Name of the destination port	
<b>Description</b>	The report_connections command will report connections between one of the following: - (-from) report all connections from a node - (-from -from_port) report connections form a port of a node - (-to) report all connections to a node - (-to -to_port) report connection to a port of a node - (-from -to) report connection between 2 nodes By default, 100 connections will be reported. In command-line mode, the result will be posted as info messages to the console. If -return_result is specified then the result will also be returned as a tcl object.		
<b>Example Usage</b>	<pre>report_connections -from my_ff report_connections -from my_ff -from_port Q report_connections -to my_ff report_connections -to my_ff -to_port D report_connections -from my_ff -to my_ff1  report_connections -from my_ff -from_port Q -limit 10 report_connections -from my_ff -from_port Q -return_result</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.20. report\_legal\_locations (::quartus::eco)

The following table displays information for the `report_legal_locations` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127		
<b>Syntax</b>	<code>report_legal_locations [-h   -help] [-long_help] [-check_routing] -location &lt;location&gt; -name &lt;node_name&gt; [-patient] [-report_illegal] [-return_result]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-check_routing</code>	Try to legalize the location by checking routing	
	<code>-location &lt;location&gt;</code>	Region to search for legal locations	
	<code>-name &lt;node_name&gt;</code>	Name of node to check legal locations for	
	<code>-patient</code>	Override restriction on search region size	
	<code>-report_illegal</code>	Report illegal locations and reasons within the search region	
	<code>-return_result</code>	Return the result in a tcl object	
<b>Description</b>	The <code>report_legal_locations</code> command will search for all legal locations within the region specified by <code>-location</code> for the specified node to be placed. The command can be used on nodes that have or have not been placed. Specify <code>-patient</code> to override the restriction on the search region size. Specify <code>-report_illegal</code> to report illegal location reasons. In command-line mode, the result will be posted as info messages to the console. If <code>-return_result</code> is specified then the result will also be returned as a tcl object. If <code>-check_routing</code> is specified then a location will be determined illegal if the legalization step fails and slacks will be reported for legal locations. Otherwise the legalization step will be skipped. Note that the <code>-check_routing</code> option is only supported when all nodes but the target have been placed. The <code>report_legal_locations</code> command does not work in "quartus_fit --eco" mode.		
<b>Example Usage</b>	<pre>report_legal_locations -name node -location "X136 Y63 X145 Y72" report_legal_locations -name node -location "X5 Y10 X5 Y10" report_legal_locations -name node -location "X5 Y10 X5 Y10" -report_illegal report_legal_locations -name node -location "X5 Y10 X5 Y10" -return_result report_legal_locations -name node -location "X5 Y10 X5 Y10" -check_routing report_legal_locations -name node -location "X136 Y63 X149 Y82" -patient report_legal_locations -name node -location "X136 Y63 X149 Y82" -patient -return_result</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.21. report\_nodes\_at\_location (::quartus::eco)

The following table displays information for the `report_nodes_at_location` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127		
<b>Syntax</b>	<code>report_nodes_at_location [-h   -help] [-long_help] -location &lt;location&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-location &lt;location&gt;</code>	location string to search	
<b>Description</b>	The <code>report_nodes_at_location</code> command reports the nodes at a given location.		
			<i>continued...</i>

<b>Example Usage</b>	report_nodes_at_location -location <location_string>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK      0      INFO: Operation successful

### 3.1.7.22. report\_ports (::quartus::eco)

The following table displays information for the `report_ports` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>report_ports [-h   -help] [-long_help] -name &lt;node_name&gt; [-return_result] [-timing]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <node_name>	Name of the node to query	
	-return_result	Return the result in a tcl object	
	-timing	Report the slacks on each port	
<b>Description</b>	The <code>report_unplaced_nodes</code> command will report all input ports of a node. In command-line mode, the result will be posted as info messages to the console. If <code>-timing</code> is specified then the slacks on each port will be reported. If <code>-return_result</code> is specified then the result will also be returned as a tcl object.		
<b>Example Usage</b>	<pre>report_ports -name my_ff report_ports -name my_ff -timing report_ports -name my_ff -return_result</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK      0      INFO: Operation successful

### 3.1.7.23. report\_routing (::quartus::eco)

The following table displays information for the `report_routing` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>report_routing [-h   -help] [-long_help] [-return_result] -to &lt;to&gt; -to_port &lt;to_port&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-return_result	Return the result in a tcl object	
	-to <to>	Name of the destination node	
	-to_port <to_port>	Name of the destination port	
<b>Description</b>	The <code>report_connections</code> command will report routing between a connection. In command-line mode, the result will be posted as info messages to the console. If <code>-return_result</code> is specified then the result will also be returned as a tcl object.		

*continued...*

<b>Example Usage</b>	report_routing -to my_ff1 -to_port D		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK 0 INFO: Operation successful

### 3.1.7.24. report\_unplaced\_nodes (::quartus::eco)

The following table displays information for the `report_unplaced_nodes` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>report_unplaced_nodes [-h   -help] [-long_help] [-return_result]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-return_result	Return the result in a tcl object	
<b>Description</b>	The <code>report_unplaced_nodes</code> command will report all unplaced nodes. In command-line mode, the result will be posted as info messages to the console. If <code>-return_result</code> is specified then the result will also be returned as a tcl object.		
<b>Example Usage</b>	<code>report_unplaced_nodes</code> <code>report_unplaced_nodes -return_result</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK 0 INFO: Operation successful

### 3.1.7.25. restore\_eco\_checkpoint (::quartus::eco)

The following table displays information for the `restore_eco_checkpoint` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::eco on page 127		
<b>Syntax</b>	<code>restore_eco_checkpoint [-h   -help] [-long_help] &lt;cid&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<cid>	Checkpoint ID of a change	
<b>Description</b>	The <code>restore_eco_checkpoint</code> command reverts the design back to the targeted checkpoint.		
<b>Example Usage</b>	<code>restore_eco_checkpoint 12345</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK 0 INFO: Operation successful

### 3.1.7.26. **unplace\_node** (::quartus::eco)

The following table displays information for the `unplace_node` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127		
<b>Syntax</b>	<code>unplace_node [-h   -help] [-long_help] -name &lt;node_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <node_name>	Name of the node to unplace	
<b>Description</b>	The <code>unplace_node</code> command will unplace a given node.		
<b>Example Usage</b>	<code>unplace_node -name my_ff</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.7.27. **update\_mif\_files** (::quartus::eco)

The following table displays information for the `update_mif_files` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::eco</a> on page 127		
<b>Syntax</b>	<code>update_mif_files [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Update memory contents from the Memory Initialization File (.mif) or Hexadecimal (Intel-Format) File (.hex) for all RAM or CAM atoms.		
<b>Example Usage</b>	<code>load_package eco update_mif_files</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

## 3.1.8. **::quartus::external\_memif\_toolkit**

The following table displays information for the **::quartus::external\_memif\_toolkit** Tcl package:

<b>Tcl Package and Version</b>	::quartus::external_memif_toolkit 1.0
<b>Description</b>	This package contains the set of Tcl functions for interacting with external memory interfaces and debug components
<b>Availability</b>	This package is available for loading in the following executables: <code>qpro_sh</code> <code>quartus_sh</code>
<b>Tcl Commands</b>	<code>apply_setting</code> ( <a href="#">::quartus::external_memif_toolkit</a> ) on page 142 <code>calibrate_termination</code> ( <a href="#">::quartus::external_memif_toolkit</a> ) on page 143 <code>configure_driver</code> ( <a href="#">::quartus::external_memif_toolkit</a> ) on page 143 <code>create_connection_report</code> ( <a href="#">::quartus::external_memif_toolkit</a> ) on page 144 <code>create_toolkit_report</code> ( <a href="#">::quartus::external_memif_toolkit</a> ) on page 145

*continued...*

```

driver_margining (::quartus::external_memif_toolkit) on page 146
establish_connection (::quartus::external_memif_toolkit) on page 147
generate_eye_diagram (::quartus::external_memif_toolkit) on page 147
get_connection_commands (::quartus::external_memif_toolkit) on page 148
get_connection_info (::quartus::external_memif_toolkit) on page 149
get_connection_interfaces (::quartus::external_memif_toolkit) on page 149
get_connection_report_info (::quartus::external_memif_toolkit) on page 150
get_connection_report_types (::quartus::external_memif_toolkit) on page 151
get_connection_types (::quartus::external_memif_toolkit) on page 152
get_connections (::quartus::external_memif_toolkit) on page 152
get_device_names (::quartus::external_memif_toolkit) on page 0
get_hardware_names (::quartus::external_memif_toolkit) on page 0
get_setting_types (::quartus::external_memif_toolkit) on page 153
get_toolkit_report_types (::quartus::external_memif_toolkit) on page 154
initialize_connections (::quartus::external_memif_toolkit) on page 154
link_project_to_device (::quartus::external_memif_toolkit) on page 155
read_setting (::quartus::external_memif_toolkit) on page 156
reindex_connections (::quartus::external_memif_toolkit) on page 157
reset_tg2 (::quartus::external_memif_toolkit) on page 157
run_connection_command (::quartus::external_memif_toolkit) on page 158
set_active_interface (::quartus::external_memif_toolkit) on page 159
set_stress_pattern (::quartus::external_memif_toolkit) on page 159
terminate_connection (::quartus::external_memif_toolkit) on page 160
terminate_connections (::quartus::external_memif_toolkit) on page 161
unlink_project_from_device (::quartus::external_memif_toolkit) on page 161
write_connection_target_report (::quartus::external_memif_toolkit) on page 162

```

### **3.1.8.1. apply\_setting (::quartus::external\_memif\_toolkit)**

The following table displays information for the apply\_setting Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141				
<b>Syntax</b>	apply_setting [-h   -help] [-long_help] -id <name> [-index <index>] [-rank <rank>] -type <type> -value <value>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-id <name>	The connection ID to communicate with			
	-index <index>	Index of the target setting within the connection			
	-rank <rank>	Rank (shadow register) of the target setting within the connection			
	-type <type>	The type of setting on the EMIF interface			
	-value <value>	Value to apply			
<b>Description</b>	Applies the specified memory interface setting for the specified target connection.				
<b>Example Usage</b>	<pre> project_open dut initialize_connections  set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     foreach type [get_setting_types -id \$conn] {         apply_setting -id \$conn -type \$type -index 0 -value 0     } } terminate_connections project_close </pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

*continued...*

TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
TCL_ERROR	1	ERROR: The specified setting type <string> is illegal. Please specify a valid setting type. The list of valid types is available by running get_setting_types.
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: The specified setting is outside the legal range.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.2. calibrate\_termination (::quartus::external\_memif\_toolkit)

The following table displays information for the calibrate\_termination Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	calibrate_termination [-h   -help] [-long_help] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
<b>Description</b>	Test termination settings on the interface.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.3. configure\_driver (::quartus::external\_memif\_toolkit)

The following table displays information for the configure\_driver Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	configure_driver [-h   -help] [-long_help] [-data <data>] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-data <data>	Address/data pairs to be written to the driver configuration block	
<i>continued...</i>			

	<code>-id &lt;name&gt;</code>		The connection ID to communicate with
<b>Description</b>	Configures the traffic generator.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <i>&lt;string&gt;</i> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.4. `create_connection_report` (::quartus::external\_memif\_toolkit)

The following table displays information for the `create_connection_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141				
<b>Syntax</b>	<code>create_connection_report [-h   -help] [-long_help] -id &lt;name&gt; -report_type &lt;name&gt;</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-id &lt;name&gt;</code>	The connection ID to communicate with			
	<code>-report_type &lt;name&gt;</code>	The report type to generate			
<b>Description</b>	Create a report of the specified type for the connection id. The resulting report is then available for query using the report TCL package.				
<b>Example Usage</b>	<pre> load_package external_memif_toolkit load_package report  project_open dut  initialize_connections  set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  set conn [lindex [get_connections -type emif] 0] establish_connection -id \$conn  create_connection_report -id \$conn -report_type summary  load_report_database -type emit  set report_panel_names [get_report_panel_names] post_message -type info "Found the following report panels:" foreach panel_name \$report_panel_names {     post_message -type info "      \$panel_name" }  unload_report  terminate_connections  project_close </pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: An error occurred trying to create the report <string> for the target <string>.
TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
TCL_ERROR	1	ERROR: The specified report type <string> is illegal. The legal report types are: <string> .
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.5. create\_toolkit\_report (::quartus::external\_memif\_toolkit)

The following table displays information for the `create_toolkit_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>create_toolkit_report [-h   -help] [-long_help] -report_type &lt;name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-report_type &lt;name&gt;</code>	The report type to generate	
<b>Description</b>	Create a toolkit report of the specified type. These reports are general toolkit reports, not connection specific reports. The resulting report is then available for query using the report TCL package.		
<b>Example Usage</b>	<pre> load_package external_memif_toolkit load_package report  project_open dut  initialize_connections  set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  create_toolkit_report -report_type discovered_connections create_toolkit_report -report_type detailed_connections  load_report_database -type emit  set report_panel_names [get_report_panel_names] post_message -type info "Found the following report panels:" foreach panel_name \$report_panel_names {     post_message -type info "      \$panel_name" }  unload_report  terminate_connections  project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: An error occurred trying to create the report <string> for the target <string>.

*continued...*

TCL_ERROR	1	ERROR: The specified report type <string> is illegal. The legal report types are: <string> .
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.6. driver\_margining (::quartus::external\_memif\_toolkit)

The following table displays information for the driver\_margining Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	driver_margining [-h   -help] [-long_help] [-adjust_delays] -fail_id <fail_id> -id <name> -pass_id <pass_id> -pnf_ids <pnf_ids> -resetn_id <resetn_id> [-skip_dm] [-skip_read] [-skip_write] [-step_size <step_size> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-adjust_delays	Adjust delays on the interface based on driver margining results	
	-fail_id <fail_id>	Connection ID of the In-System Probe which controls the fail signal from the driver	
	-id <name>	The connection ID to communicate with	
	-pass_id <pass_id>	Connection ID of the In-System Probe which controls the pass signal from the driver	
	-pnf_ids <pnf_ids>	TCL list of PNF (pass not fail) connection IDs of In-System Probes from the driver	
	-resetn_id <resetn_id>	Connection ID of the In-System Source which controls the resetn for the driver	
	-skip_dm	Skip driver margining on dm	
	-skip_read	Skip driver margining on read	
<b>Description</b>	Performs read and write driver margining on the selected EMIF connection.		
	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.7. establish\_connection (::quartus::external\_memif\_toolkit)

The following table displays information for the establish\_connection Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	establish_connection [-h   -help] [-long_help] [-connection_name <name>] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-connection_name <name>	Optional nickname to use for a connection.	
	-id <name>	The connection ID to communicate with	
<b>Description</b>	Establishes a connection to a connection target.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof set conn [lindex [get.connections -type emif] 0] establish_connection -id \$conn terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Could not establish a connection to target <string>.
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The specified connection name <string> is illegal as a connection already exists with that name. Please specify a valid connection name.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.8. generate\_eye\_diagram (::quartus::external\_memif\_toolkit)

The following table displays information for the generate\_eye\_diagram Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	generate_eye_diagram [-h   -help] [-long_help] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	

*continued...*

	-id <name>		The connection ID to communicate with
<b>Description</b>	Generates eye diagrams for the selected EMIF connection.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.9. get\_connection\_commands (::quartus::external\_memif\_toolkit)

The following table displays information for the get\_connection\_commands Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	get_connection_commands [-h   -help] [-long_help] -id <name>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-id <name>		The connection ID to communicate with
<b>Description</b>	Returns a TCL list of supported commands for the specific target connection.		
<b>Example Usage</b>	<pre> project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     puts "Connection : \$conn"     foreach cmd [get_connection_commands -id \$conn] {         puts "    Command : \$cmd"     } } terminate_connections project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.10. get\_connection\_info (::quartus::external\_memif\_toolkit)

The following table displays information for the `get_connection_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>get_connection_info [-h   -help] [-long_help] [-hpath] -id &lt;name&gt; [-sld_type]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-hpath</code>	Queries the hierarchy name of the connection	
	<code>-id &lt;name&gt;</code>	The connection ID to communicate with	
	<code>-sld_type</code>	Queries SLD type of a connection	
<b>Description</b>	Queries information about a connection target.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections  set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     puts "Connection : \$conn"     puts "Hierarchy Path : [get_connection_info -id \$conn -hpath]" }  terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: No information about the connection ID was queried. Please specify one query parameter.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.11. get\_connection\_interfaces (::quartus::external\_memif\_toolkit)

The following table displays information for the `get_connection_interfaces` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>get_connection_interfaces [-h   -help] [-long_help] -id &lt;name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-id &lt;name&gt;</code>	The connection ID to communicate with	
<b>Description</b>	Returns the interfaces available for the specified connection id.		
<i>continued...</i>			

<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  set conn [lindex [get.connections -type emif] 0] establish_connection -id \$conn get_connection_interfaces -id \$conn terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.12. get\_connection\_report\_info (::quartus::external\_memif\_toolkit)

The following table displays information for the get\_connection\_report\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141	
<b>Syntax</b>	get_connection_report_info [-h   -help] [-long_help] -id <name> [-name] -report_type <name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-id <name>	The connection ID to communicate with
	-name	Queries the name of the report
	-report_type <name>	The report type to generate
<b>Description</b>	Queries info about a report type.	
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get.connections] {     puts "Connection : \$conn"     foreach rpt [get.connection_report_types -id \$conn] {         puts "    Report : \$rpt"         puts "    Report name : [get_connection_report_info -id \$conn -report_type \$rpt -name]"     } } terminate_connections project_close</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The specified report type <string> is illegal. The legal report types are: <string> .
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.13. get\_connection\_report\_types (::quartus::external\_memif\_toolkit)

The following table displays information for the get\_connection\_report\_types Tcl command:

Tcl Package and Version	Belongs to ::quartus::external_memif_toolkit on page 141		
Syntax	get_connection_report_types [-h   -help] [-long_help] -id <name>		
Arguments	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
Description	Returns a TCL list of supported report types for the specific target connection.		
Example Usage	<pre> project_open dut initialize_connections  set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     puts "Connection : \$conn"     puts "Supported report types: [get_connection_report_types -id \$conn]" }  terminate_connections project_close </pre>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.14. `get_connection_types` (::quartus::external\_memif\_toolkit)

The following table displays information for the `get_connection_types` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>get_connection_types [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns a list of valid connection target types.		
<b>Example Usage</b>	<pre>puts "Valid connection types are: [get_connection_types]"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.8.15. `get_connections` (::quartus::external\_memif\_toolkit)

The following table displays information for the `get_connections` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>get_connections [-h   -help] [-long_help] [-type &lt;name&gt; ]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-type &lt;name&gt;</code>	The type of the connection to connect to	
<b>Description</b>	Returns a TCL list of connection IDs for connections.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof foreach conn [get_connections] {     puts "Connection : \$conn"     puts "    Hierarchy Path : [get_connection_info -id \$conn -hpath]" } terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Supplying hardware_name and device_name is mutually exclusive to supplying type
	TCL_ERROR	1	ERROR: Both hardware_name and device_name must be supplied if either is supplied
	TCL_ERROR	1	ERROR: Connection type <string> is not a recognized connection type.

*continued...*

TCL_ERROR	1	ERROR: No device name called <string> could be found on hardware named <string>.
TCL_ERROR	1	ERROR: No hardware name called <string> could be found.
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.16. get\_setting\_types (::quartus::external\_memif\_toolkit)

The following table displays information for the get\_setting\_types Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	get_setting_types [-h   -help] [-long_help] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
<b>Description</b>	Returns a TCL list of supported memory interface setting types for the specific target connection.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     puts "Connection : \$conn"     puts "Supported setting types: [get_setting_types -id \$conn]" } terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.17. get\_toolkit\_report\_types (::quartus::external\_memif\_toolkit)

The following table displays information for the get\_toolkit\_report\_types Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	get_toolkit_report_types [-h   -help] [-long_help]		
<i>continued...</i>			

<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
<b>Description</b>	Returns a TCL list of supported toolkit report types. These reports are general toolkit reports, not connection specific reports.		
<b>Example Usage</b>	<pre> project_open dut initialize_connections  set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof puts "Supported toolkit report types: [get_toolkit_report_types]"  terminate_connections project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.18. initialize\_connections (::quartus::external\_memif\_toolkit)

The following table displays information for the initialize\_connections Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141						
<b>Syntax</b>	initialize_connections [-h   -help] [-long_help]						
<b>Arguments</b>	<table border="1"> <tr> <td>-h   -help</td> <td>Short help</td> </tr> <tr> <td>-long_help</td> <td>Long help with examples and possible return values</td> </tr> </table>			-h   -help	Short help	-long_help	Long help with examples and possible return values
-h   -help	Short help						
-long_help	Long help with examples and possible return values						
<b>Description</b>	Initializes the internal data structures of the toolkit. This command must be run before any other toolkit commands are executed.						
<b>Example Usage</b>	<pre> project_open dut initialize_connections terminate_connections project_close </pre>						
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>				
	TCL_OK	0	INFO: Operation successful				
	TCL_ERROR	1	ERROR: Unable to find active revision. Specify an active revision name using set_current_revision <revision name>.				
	TCL_ERROR	1	ERROR: The current project specifies a part that cannot be loaded. Verify the device family is correctly installed.				
	TCL_ERROR	1	ERROR: The connection to the hardware drivers could not be established.				
	TCL_ERROR	1	ERROR: Illegal connections detected				

*continued...*

TCL_ERROR	1	ERROR: The current project settings are invalid. Verify that all project settings are valid.
TCL_ERROR	1	ERROR: The JTAG Debug Information (.jdi) file called <string> could not be found. Ensure this file exists or run quartus_asm to create it.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Could not initialize System Console. <string>
TCL_ERROR	1	ERROR: Toolkit has already been initialized. Use terminate_connections to de-initialize the toolkit.

### 3.1.8.19. link\_project\_to\_device (::quartus::external\_memif\_toolkit)

The following table displays information for the link\_project\_to\_device Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	link_project_to_device [-h   -help] [-long_help] -device_name <name> -hardware_name <name> [-jdi_file <name>] [-sof_file <name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-device_name <name>	The name of the device to connect to	
	-hardware_name <name>	The name of the hardware connection to use	
	-jdi_file <name>	Specifies the JTAG Debugging Information file to use when creating the design link.	
	-sof_file <name>	Specifies the SOF file to use when creating the design link.	
<b>Description</b>	Links the currently opened project to the specified target device on the specified hardware.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: An error occurred while linking the project to the device.
	TCL_ERROR	1	ERROR: No device name called <string> could be found on hardware named <string>.
	TCL_ERROR	1	ERROR: No hardware name called <string> could be found.
	TCL_ERROR	1	ERROR: The JTAG Debug Information (.jdi) file called <string> could not be found. Ensure this file exists or run quartus_asm to create it.

**continued...**

TCL_ERROR	1	ERROR: A JTAG Debug Information (.jdi) file or a SOF file (.sof) is required for linking a project to a device, but no file was supplied. Ensure this file exists or run quartus_asm to create it.
TCL_ERROR	1	ERROR: The currently opened project has already been linked to a device. Use unlink_project_from_device to unlink the project from a device.
TCL_ERROR	1	ERROR: The SOF (.sof) file called <string> could not be found. Ensure this file exists or run quartus_asm to create it.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.20. read\_setting (::quartus::external\_memif\_toolkit)

The following table displays information for the read\_setting Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	read_setting [-h   -help] [-long_help] -id <name> [-index <index> ] [-rank <rank> ] -type <type>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
	-index <index>	Index of the target setting within the connection	
	-rank <rank>	Rank (shadow register) of the target setting within the connection	
	-type <type>	The type of setting on the EMIF interface	
<b>Description</b>	Reads the specified memory interface setting for the specified target connection.		
<b>Example Usage</b>	<pre> project_open dut initialize_connections  set hw_name [lindex [get.hardware_names] 0] set dev_name [lindex [get.device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  foreach conn [get_connections] {     foreach type [get_setting_types -id \$conn] {         read_setting -id \$conn -type \$type -index 0     } } terminate_connections project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The specified setting type <string> is illegal. Please specify a valid setting type. The list of valid types is available by running get_setting_types.

*continued...*

TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: The specified setting is outside the legal range.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.21. reindex\_connections (::quartus::external\_memif\_toolkit)

The following table displays information for the `reindex_connections` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>reindex_connections [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Reinitializes the list of known hardware connections.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections foreach hw_name [get_hardware_names] {     puts "Found hardware name \$hw_name" } reindex_connections foreach hw_name [get_hardware_names] {     puts "Found hardware name \$hw_name" } terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.22. reset\_tg2 (::quartus::external\_memif\_toolkit)

The following table displays information for the `reset_tg2` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>reset_tg2 [-h   -help] [-long_help] -resetn_id &lt;resetn_id&gt; -tg2_id &lt;tg2_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-resetn_id &lt;resetn_id&gt;</code>	Connection ID of the resetn	
	<code>-tg2_id &lt;tg2_id&gt;</code>	Connection ID of the Traffic Generator 2.0 associated with this interface	
<b>Description</b>	Resets a Traffic Generator 2.0 instance.		
	<i>continued...</i>		

<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.23. run\_connection\_command (::quartus::external\_memif\_toolkit)

The following table displays information for the run\_connection\_command Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	run_connection_command [-h   -help] [-long_help] -command_name <name> -id <name> [-payload <name> ]		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-command_name <name>		The command name to execute
	-id <name>		The connection ID to communicate with
	-payload <name>		TCL list of parameter data to use when executing the command
<b>Description</b>	Executes a command of the specified type and for the connection id.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections  set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  set conn [lindex [get_connections -type emif] 0] establish_connection -id \$conn  run_connection_command -id \$conn -command_name nop  terminate_connections  project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: An error occurred trying to execute the command <string> for the target <string>.
	TCL_ERROR	1	ERROR: The specified command name <string> is illegal. The legal command names are: <string> .
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.

*continued...*

TCL_ERROR	1	ERROR: No information about the report was queried. Please specify one query parameter.
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.24. set\_active\_interface (::quartus::external\_memif\_toolkit)

The following table displays information for the `set_active_interface` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	<code>set_active_interface [-h   -help] [-long_help] -id &lt;name&gt; -interface_id &lt;name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
	-interface_id <name>	The interface ID to communicate with	
<b>Description</b>	Selects the active interface to debug for the specified connection id.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The specified interface ID <string> is illegal. Please specify a valid interface ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.25. set\_stress\_pattern (::quartus::external\_memif\_toolkit)

The following table displays information for the `set_stress_pattern` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141	
<b>Syntax</b>	<code>set_stress_pattern [-h   -help] [-long_help] -id &lt;name&gt; -value &lt;value&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-id <name>	The connection ID to communicate with
	-value <value>	1 to enable stress pattern in calibration, 0 to disable

**continued...**

<b>Description</b>	Enable or disable the stress pattern for future calibrations.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.26. terminate\_connection (::quartus::external\_memif\_toolkit)

The following table displays information for the terminate\_connection Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	terminate_connection [-h   -help] [-long_help] -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <name>	The connection ID to communicate with	
<b>Description</b>	Terminates a connection to a connection target.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof         set conn [lindex [get_connections -type emif] 0]         establish_connection -id \$conn         terminate_connection -id \$conn         terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Connection ID <string> could not be terminated because the connection was not established.
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.27. terminate\_connections (::quartus::external\_memif\_toolkit)

The following table displays information for the terminate\_connections Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	terminate_connections [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Deletes the internal data structures of the toolkit.		
<b>Example Usage</b>	<pre>project_open dut puts "Preparing to initialize connections" initialize_connections puts "Preparing to delete connections" terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.28. unlink\_project\_from\_device (::quartus::external\_memif\_toolkit)

The following table displays information for the unlink\_project\_from\_device Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	unlink_project_from_device [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Unlinks the currently opened project from the currently linked target device on the specified hardware.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof unlink_project_from_device link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: The JTAG Debug Information (.jdi) file called <string> could not be found. Ensure this file exists or run quartus_asm to create it.
TCL_ERROR	1	ERROR: The currently opened project has not been linked to a device. Run link_project_to_device to link a project to a device.
TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.8.29. write\_connection\_target\_report (::quartus::external\_memif\_toolkit)

The following table displays information for the write\_connection\_target\_report Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::external_memif_toolkit on page 141		
<b>Syntax</b>	write_connection_target_report [-h   -help] [-long_help] -file <name> -id <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-file <name>	File name of report to write	
	-id <name>	The connection ID to communicate with	
<b>Description</b>	Writes the reports for the given connection target to a filename.		
<b>Example Usage</b>	<pre>project_open dut initialize_connections  set hw_name [lindex [get_hardware_names] 0] set dev_name [lindex [get_device_names -hardware_name \$hw_name] 0] link_project_to_device -hardware_name \$hw_name -device_name \$dev_name -sof_file dut.sof  set conn [lindex [get_connections] 0] write_connection_target_report -id \$conn -file report.rpt  terminate_connections project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified connection ID <string> is illegal. Please specify a valid connection ID.
	TCL_ERROR	1	ERROR: No reports for connection ID <string> have been generated. Please run create_report_for_target before writing reports to file.
	TCL_ERROR	1	ERROR: Toolkit has not been initialized. Use initialize_connections to initialize the toolkit.

### 3.1.9. ::quartus::fif

The following table displays information for the ::quartus::fif Tcl package:

<b>Tcl Package and Version</b>	::quartus::fif 1.0
<b>Description</b>	This package contains the set of Tcl functions for using the Fault Injection File (FIF) Driver.
<i>continued...</i>	

<b>Availability</b>	This package is loaded by default in the following executable: <code>quartus_fif</code>
<b>Tcl Commands</b>	<code>check (::quartus::fifo)</code> on page 163 <code>dump (::quartus::fifo)</code> on page 163 <code>dump_cram_frame (::quartus::fifo)</code> on page 164 <code>dump_mem (::quartus::fifo)</code> on page 164 <code>dump_pr_bitstream (::quartus::fifo)</code> on page 165 <code>generate (::quartus::fifo)</code> on page 165 <code>get_frame_count (::quartus::fifo)</code> on page 166 <code>get_frame_size (::quartus::fifo)</code> on page 166 <code>get_sector_information_sdm_based_fpga (::quartus::fifo)</code> on page 167 <code>get_sensitive_location (::quartus::fifo)</code> on page 167 <code>get_sensitive_location_sdm_based_fpga (::quartus::fifo)</code> on page 168 <code>setup_sdm_based_fpga (::quartus::fifo)</code> on page 168 <code>terminate (::quartus::fifo)</code> on page 169

### 3.1.9.1. check (::quartus::fifo)

The following table displays information for the `check` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::fifo</code> on page 162		
<b>Syntax</b>	<code>check [-h   -help] [-long_help] -frame &lt;frame&gt; -index &lt;index&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-frame &lt;frame&gt;</code>	CRAM frame ID	
	<code>-index &lt;index&gt;</code>	CRAM frame bit location	
<b>Description</b>	Check if the specified location contains sensitive bit. Returns 1, if the specified location contains sensitive bit. Returns 0, otherwise.		
<b>Example Usage</b>	<code>check -frame 3 -index 100</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	<code>TCL_OK</code>	0	INFO: Operation successful
	<code>TCL_ERROR</code>	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.2. dump (::quartus::fifo)

The following table displays information for the `dump` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::fifo</code> on page 162		
<b>Syntax</b>	<code>dump [-h   -help] [-long_help] [-all]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-all</code>	Option to display all information	
	Dump FIF driver contents.		
<b>Example Usage</b>	<code>dump</code> <code>dump -all</code>		
<i>continued...</i>			

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.3. dump\_cram\_frame (::quartus::fifo)

The following table displays information for the `dump_cram_frame` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>dump_cram_frame [-h   -help] [-long_help] [-all] -frame &lt;frame&gt;</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-all		Option to display all information
	-frame <frame>		CRAM Frame ID
<b>Description</b>	Dump CRAM frame information.		
<b>Example Usage</b>	<code>dump_cram_frame -frame 3</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.4. dump\_mem (::quartus::mem)

The following table displays information for the `dump_mem` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::mem on page 162		
<b>Syntax</b>	<code>dump_mem [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
<b>Description</b>	Dump memory usage.		
<b>Example Usage</b>	<code>dump_mem</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.9.5. dump\_pr\_bitstream (::quartus::fifo)

The following table displays information for the `dump_pr_bitstream` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>dump_pr_bitstream [-h   -help] [-long_help] -id &lt;id&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <id>	Static PR Bitstream ID	
<b>Description</b>	Dump static PR bitstream.		
<b>Example Usage</b>	<code>dump_pr_bitstream -id 3</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.
	TCL_ERROR	1	ERROR: Parameter <string> has exceeded range.
	TCL_ERROR	1	ERROR: <string> mismatch. <string>

### 3.1.9.6. generate (::quartus::fifo)

The following table displays information for the `generate` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>generate [-h   -help] [-long_help] [-error_count &lt;error_count&gt;] [-error_index &lt;error_index&gt;] -frame &lt;frame&gt; -output &lt;output&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-error_count <error_count>	Number of errors to be injected	
	-error_index <error_index>	CRAM frame error bit locations	
	-frame <frame>	CRAM frame ID	
	-output <output>	Output file name	
<b>Description</b>	Generates PR RBF file. Specifies the 'error_count' and 'error_index' to generate a PR RBF file with fault injected. Otherwise, the command will generate a PR RBF file without fault for external scrubbing.		
<b>Example Usage</b>	<code>generate -frame 3 -output scrub.rbf generate -frame 3 -output inject.rbf -error_count 1 -error_index 100 generate -frame 3 -output inject.rbf -error_count 2 -error_index 100 200</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.

*continued...*

TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.
TCL_ERROR	1	ERROR: Missing parameter <string>.
TCL_ERROR	1	ERROR: Parameter <string> has exceeded range.
TCL_ERROR	1	ERROR: <string> mismatch. <string>
TCL_ERROR	1	ERROR: Error writing to file (<string>).

### 3.1.9.7. get\_frame\_count (::quartus::fifo)

The following table displays information for the get\_frame\_count Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	get_frame_count [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns total frame count.		
<b>Example Usage</b>	get_frame_count		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.8. get\_frame\_size (::quartus::fifo)

The following table displays information for the get\_frame\_size Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	get_frame_size [-h   -help] [-long_help] -frame <frame>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-frame <frame>	CRAM frame ID	
<b>Description</b>	Returns frame size of the specific frame.		
<b>Example Usage</b>	get_frame_size -frame 3		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.9. get\_sector\_information\_sdm\_based\_fpga (::quartus::fifo)

The following table displays information for the get\_sector\_information\_sdm\_based\_fpga Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	get_sector_information_sdm_based_fpga [-h   -help] [-long_help] -sector_index <sector_index>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-sector_index <sector_index>	Sector Index	
<b>Description</b>	Query information about sector with given index number.		
<b>Example Usage</b>	get_sector_information_sdm_based_fpga -sector_index 25		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.
	TCL_ERROR	1	ERROR: Missing parameter <string>.

### 3.1.9.10. get\_sensitive\_location (::quartus::fifo)

The following table displays information for the get\_sensitive\_location Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	get_sensitive_location [-h   -help] [-long_help] [-count <count>] -frame <frame>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-count <count>	Option to return specific number of indexes	
	-frame <frame>	CRAM frame ID	
<b>Description</b>	Returns a list of sensitive locations of the specific frame.		
<b>Example Usage</b>	get_sensitive_location -frame 3 get_sensitive_location -frame 3 -count 10		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.11. get\_sensitive\_location\_sdm\_based\_fpga (::quartus::fifo)

The following table displays information for the `get_sensitive_location_sdm_based_fpga` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>get_sensitive_location_sdm_based_fpga [-h   -help] [-long_help] -bit_index &lt;bit_index&gt; -frame_index &lt;frame_index&gt; -sector_index &lt;sector_index&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-bit_index <bit_index>	Bit Index	
	-frame_index <frame_index>	Frame Index	
	-sector_index <sector_index>	Sector Index	
<b>Description</b>	Check if the bit is sensitive or not at specific location.		
<b>Example Usage</b>	<code>get_sensitive_location_sdm_based_fpga -sector_index 25 -frame_index 4 -bit_index 3</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has not been setup. Use setup command to setup the FIF driver.

### 3.1.9.12. setup (::quartus::fifo)

The following table displays information for the `setup` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>setup [-h   -help] [-long_help] -fifo &lt;fifo&gt; -rbf &lt;rbf&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fifo <fifo>	FIF filename	
	-rbf <rbf>	RBF filename	
<b>Description</b>	Setup FIF driver.		
<b>Example Usage</b>	<code>setup -fifo test.fif -rbf test.rbf</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIF driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIF driver has already been setup.

### 3.1.9.13. **setup\_sdm\_based\_fpga** (::quartus::fifo)

The following table displays information for the `setup_sdm_based_fpga` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>setup_sdm_based_fpga [-h   -help] [-long_help] -fifo &lt;fifo&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fifo <fifo>	FIFO filename	
<b>Description</b>	Setup FIFO driver for SDM based FPGA.		
<b>Example Usage</b>	<code>setup_sdm_based_fpga -fifo test.fifo</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIFO driver operation failed with error <string>.
	TCL_ERROR	1	ERROR: FIFO driver has already been setup.
	TCL_ERROR	1	ERROR: FIFO file is not compatible.

### 3.1.9.14. **terminate** (::quartus::fifo)

The following table displays information for the `terminate` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fifo on page 162		
<b>Syntax</b>	<code>terminate [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Terminate FIFO driver.		
<b>Example Usage</b>	<code>terminate</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FIFO driver has not been setup. Use setup command to setup the FIFO driver.

## 3.1.10. ::quartus::flng

The following table displays information for the ::quartus::flng Tcl package:

<b>Tcl Package and Version</b>	::quartus::flng 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is loaded by default in the following executables: <code>qpro</code> <code>qpro_sh</code>

*continued...*

	<pre> quartus quartus_cdb quartus_da quartus_edt quartus_fit quartus_ipgenerate quartus_map quartus_sh quartus_syn quartus_tlg qunb </pre>
<b>Tcl Commands</b>	<p><a href="#">flng::add_object (::quartus::flng)</a> on page 170  <a href="#">flng::add_property (::quartus::flng)</a> on page 171  <a href="#">flng::bind_flow (::quartus::flng)</a> on page 171  <a href="#">flng::delete_object (::quartus::flng)</a> on page 172  <a href="#">flng::get_flow_list (::quartus::flng)</a> on page 172  <a href="#">flng::get_next_available_id (::quartus::flng)</a> on page 172  <a href="#">flng::get_object (::quartus::flng)</a> on page 173  <a href="#">flng::get_objects (::quartus::flng)</a> on page 173  <a href="#">flng::get_option (::quartus::flng)</a> on page 174  <a href="#">flng::get_property (::quartus::flng)</a> on page 174  <a href="#">flng::get_task_command (::quartus::flng)</a> on page 175  <a href="#">flng::init_repository (::quartus::flng)</a> on page 175  <a href="#">flng::list_properties (::quartus::flng)</a> on page 175  <a href="#">flng::monitor_flow (::quartus::flng)</a> on page 176  <a href="#">flng::run_flow (::quartus::flng)</a> on page 176  <a href="#">flng::set_option (::quartus::flng)</a> on page 177  <a href="#">flng::set_property (::quartus::flng)</a> on page 177</p>

### 3.1.10.1. flng::add\_object (::quartus::flng)

The following table displays information for the `flng::add_object` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::flng</a> on page 169		
<b>Syntax</b>	<code>flng::add_object [-h   -help] [-long_help] [-name &lt;name&gt;] [-number &lt;number&gt;] -type &lt;type&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-name &lt;name&gt;</code>	The name of the new flow or task object.	
	<code>-number &lt;number&gt;</code>	The numeric part of the new FLNG object's name.	
	<code>-type &lt;type&gt;</code>	The type of the new FLNG object.	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to add object.
	TCL_ERROR	1	ERROR: Invalid object type '<string>'.
	TCL_ERROR	1	ERROR: Object name is needed.
	TCL_ERROR	1	ERROR: Object number is needed.

### 3.1.10.2. flng::add\_property (::quartus::flng)

The following table displays information for the flng::add\_property Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::add_property [-h   -help] [-long_help] -name <name> -object <object> -value <value> [-value_type <value_type> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The object to which the new property belongs.	
	-value <value>	property value.	
	-value_type <value_type>	property value type.	
<b>Description</b>	Add a property (a name/value pair) to a generic object		
<b>Example Usage</b>	flng::add_property -object <object> -name <property name> -value <property value> -value_type <value type>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to add object property.
	TCL_ERROR	1	ERROR: Invalid object '%'s.
	TCL_ERROR	1	ERROR: Invalid property value.
	TCL_ERROR	1	ERROR: Invalid property value type.
	TCL_ERROR	1	ERROR: Property '<string>' already exists. Cannot be added.

### 3.1.10.3. flng::bind\_flow (::quartus::flng)

The following table displays information for the flng::bind\_flow Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::bind_flow [-h   -help] [-long_help] [-end <end>] -flow <flow> [-start <start> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-end <end>	The name of the last task to bind in a flow.	
	-flow <flow>	The name of the flow to bind qsf.	
	-start <start>	The name of the first task to bind in a flow	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	flng::bind_flow -flow flowname		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.10.4. flng::delete\_object (::quartus::flng)

The following table displays information for the flng::delete\_object Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::delete_object [-h   -help] [-long_help] -object <object>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-object <object>	Object to delete	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to delete object.

### 3.1.10.5. flng::get\_flow\_list (::quartus::flng)

The following table displays information for the flng::get\_flow\_list Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::get_flow_list [-h   -help] [-long_help] -project <project> -project_path <project_path> -revision <revision>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-project <project>	The name of the project	
	-project_path <project_path>	The project path	
	-revision <revision>	The name of the revision	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	flng::get_flow_list -project project -revision revision		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.10.6. flng::get\_next\_available\_id (::quartus::flng)

The following table displays information for the flng::get\_next\_available\_id Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::get_next_available_id [-h   -help] [-long_help] -type <type>		

*continued...*

<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-type <type>		The type of the FLNG object.
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to get next available id
	TCL_ERROR	1	ERROR: Invalid object type '<string>'.

### 3.1.10.7. flng::get\_object (::quartus::flng)

The following table displays information for the flng::get\_object Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::get_object [-h   -help] [-long_help] [-name <name>] [-number <number>] [-properties <properties>] -type <type>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-name <name>		The literal name of the objects to be retrieved.
	-number <number>		The numerical part of the name of the objects to be retrieved.
	-properties <properties>		Find object matching the specified properties.
	-type <type>		The type of objects to be retrieved.
<b>Description</b>	Find a flow engine object matching type and name.		
<b>Example Usage</b>	flng::get_object -type task -name synthesis		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Fail to get object.

### 3.1.10.8. flng::get\_objects (::quartus::flng)

The following table displays information for the flng::get\_objects Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::get_objects [-h   -help] [-long_help] [-name <name>] [-number <number>] [-properties <properties>] -type <type>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-name <name>		The literal name of the objects to be retrieved.

*continued...*

	-number <number>	The numerical part of the name of the objects to be retrieved.
	-properties <properties>	Include only objects that the specified properties
	-type <type>	The type of objects to be retrieved.
<b>Description</b>	Get a list of specific existing Flow Engine objects	
<b>Example Usage</b>	<code>flng::get_objects -type task</code>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	TCL_ERROR	1
	<b>String Return</b>	
	INFO: Operation successful	
	ERROR: Fail to get objects.	

### 3.1.10.9. `flng::get_objects` (::quartus::fng)

The following table displays information for the `flng::get_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fng on page 169		
<b>Syntax</b>	<code>flng::get_option [-h   -help] [-long_help] -name &lt;name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	option name.	
<b>Description</b>	Get option for the Flow Engine system		
<b>Example Usage</b>	<code>flng::get_option -name &lt;option name&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FLOW option '<string>' is invalid.

### 3.1.10.10. `flng::get_property` (::quartus::fng)

The following table displays information for the `flng::get_property` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::fng on page 169		
<b>Syntax</b>	<code>flng::get_property [-h   -help] [-long_help] -name &lt;name&gt; -object &lt;object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The object to which the property belongs.	
<b>Description</b>	Get the value of a generic object's property.		
<b>Example Usage</b>	<code>flng::get_property -object &lt;object&gt; -name &lt;property name&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	continued...		

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: Invalid object '%'s.
TCL_ERROR	1	ERROR: Property '<string>' is invalid.

### 3.1.10.11. flng::get\_task\_command (::quartus::flng)

The following table displays information for the flng::get\_task\_command Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::get_task_command [-h   -help] [-long_help] -task_instance <task_instance>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-task_instance <task_instance>	The name of the task_instance	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	flng::get_task_command -task_instance task_instance		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.10.12. flng::init\_repository (::quartus::flng)

The following table displays information for the flng::init\_repository Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::init_repository [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	flng::init_repository		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.10.13. flng::list\_properties (::quartus::flng)

The following table displays information for the flng::list\_properties Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	flng::list_properties [-h   -help] [-long_help] -object <object>		
<b>Arguments</b>	-h   -help	Short help	

*continued...*

	-long_help	Long help with examples and possible return values			
	-object <object>	The object whose properties are returned as a list of property names.			
<b>Description</b>	Get the list of property names from a generic object				
<b>Example Usage</b>	<code>flng::list_properties -object &lt;object&gt;</code>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Invalid object '%s'.		

### 3.1.10.14. flng::monitor\_flow (::quartus::flng)

The following table displays information for the `flng::monitor_flow` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::flng</a> on page 169		
<b>Syntax</b>	<code>flng::monitor_flow [-h   -help] [-long_help] -flow &lt;flow&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-flow <flow>	The name of the flow to run.	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<code>flng::monitor_flow -flow flowname</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.10.15. flng::run\_flow (::quartus::flng)

The following table displays information for the `flng::run_flow` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::flng</a> on page 169		
<b>Syntax</b>	<code>flng::run_flow [-h   -help] [-long_help] [-end &lt;end&gt;] -flow &lt;flow&gt; [-print_only] [-resume] [-start &lt;start&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-end <end>	The name of the last task to run in a flow.	
	-flow <flow>	The name of the flow to run.	
	-print_only	Print what will be run and do not execute.	
	-resume	Resume executing flow from where it left off.	
	-start <start>	The name of the first task to run in a flow	
<b>Description</b>	This command currently contains no help description.		

*continued...*

<b>Example Usage</b>	<code>flng::run_flow -flow flowname</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK      0      INFO: Operation successful

### 3.1.10.16. flng::set\_option (::quartus::flng)

The following table displays information for the `fng::set_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	<code>fng::set_option [-h   -help] [-long_help] -name &lt;name&gt; -value &lt;value&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	option name.	
	-value <value>	option value.	
<b>Description</b>	Set options for the Flow Engine system		
<b>Example Usage</b>	<code>fng::set_option -name &lt;option name&gt; -value &lt;option value&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: FLOW option '<string>' is invalid.

### 3.1.10.17. flng::set\_property (::quartus::flng)

The following table displays information for the `fng::set_property` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flng on page 169		
<b>Syntax</b>	<code>fng::set_property [-h   -help] [-long_help] -name &lt;name&gt; -object &lt;object&gt; -value &lt;value&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	property name.	
	-object <object>	The object to which the property belongs.	
	-value <value>	property value.	
<b>Description</b>	Update property in term of name/value pair to a generic flow object		
<b>Example Usage</b>	<code>fng::set_property -object &lt;object&gt; -name &lt;property name&gt; -value &lt;property value&gt;</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid object '%'.s.
	TCL_ERROR	1	ERROR: Property '<string>' is invalid.

### 3.1.11. ::quartus::flow

The following table displays information for the **::quartus::flow** Tcl package:

<b>Tcl Package and Version</b>	::quartus::flow 1.1
<b>Description</b>	This package contains the set of Tcl functions for running flows or command-line executables.
<b>Availability</b>	<p>This package is available for loading in the following executables:</p> <pre> hdb_debug qpro qpro_sh quartus quartus_cdb quartus_drc quartus_eda quartus_fit quartus_ipgenerate quartus_map quartus_sh quartus_si quartus_sim quartus_sta quartus_stp quartus_syn quartus_tlg </pre>
<b>Tcl Commands</b>	<a href="#">execute_flow</a> (:quartus::flow) on page 178 <a href="#">execute_module</a> (:quartus::flow) on page 180 <a href="#">get_flow_templates</a> (:quartus::flow) on page 181 <a href="#">write_flow_assignment_digest</a> (:quartus::flow) on page 181 <a href="#">write_flow_finished</a> (:quartus::flow) on page 182 <a href="#">write_flow_started</a> (:quartus::flow) on page 182 <a href="#">write_flow_template</a> (:quartus::flow) on page 183

#### 3.1.11.1. execute\_flow (:quartus::flow)

The following table displays information for the `execute_flow` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::flow</a> on page 178	
<b>Syntax</b>	<code>execute_flow [-h   -help] [-long_help] [-analysis_and_elaboration] [-check_ios] [-check_netlist] [-compile] [-dont_export_assignments] [-eco &lt;value&gt;] [-export_database] [-finalize] [-generate_functional_sim_netlist] [-implement] [-import_database] [-incremental_compilation_export] [-incremental_compilation_import] [-quick_elaboration] [-signalprobe]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-analysis_and_elaboration</code>	Option to run Analysis and Elaboration
	<code>-check_ios</code>	Option to run I/O assignment analysis
	<code>-check_netlist</code>	Option to run Check and Save Netlist
	<code>-compile</code>	Option to run a full compilation
	<code>-dont_export_assignments</code>	Option not to export assignments to file. By default, this command exports assignments before running command-line executables.
	<code>-eco &lt;value&gt;</code>	Option to run a Fitter ECO compilation
	<code>-export_database</code>	Option to export a version-compatible database

*continued...*

	-finalize	Option to run algorithms to prepare design for programming.	
	-generate_functional_sim_netlist	Option to generate a functional simulation netlist	
	-implement	Option to run compilation up to route stage and skipping all time intensive algorithms after.	
	-import_database	Option to import a version-compatible database	
	-incremental_compilation_export	Option to export a design partition into a Quartus Prime Exported Partition (QXP) file	
	-incremental_compilation_import	Option to import one or more Quartus Prime Exported Partition (QXP) files into the design partitions of the current project	
	-quick_elaboration	Option to run Quick Elaboration	
	-signalprobe	Option to run a Signal Probe compilation	
<b>Description</b>	<p>Runs one or more of the command-line executables using one of the predefined flows, such as "-compile" or "-signalprobe". You can run only one flow at a time, so you must use only one option. Some flows have limited device support or other limitations based on the features used. See documentation for the features in question for details. The "-export_database" and "-import_database" options use the value of the VER_COMPATIBLE_DB_DIR assignment for the version-compatible database files directory, defaulting to "export_db". The "-incremental_compilation_export" option uses the value of the INCREMENTAL_COMPILATION_EXPORT_FILE global assignment for the path of the Quartus Prime Exported Partition (QXP) file to be created. The value of the INCREMENTAL_COMPILATION_EXPORT_PARTITION_NAME global assignment should specify the name of the partition to be exported. The value of the INCREMENTAL_COMPILATION_EXPORT_NELIST_TYPE global assignment (which can either have value POST_SYNTH or POST_FIT) determines whether post-synthesis or post-fitting results should be exported. Finally, the value of the INCREMENTAL_COMPILATION_EXPORT_ROUTING global assignment specifies whether routing should be exported when a post-fit netlist is generated. The "-incremental_compilation_import" option uses the following partition assignments to determine the location of the QXP files, and how importation should be performed, on a per-partition basis:</p> <p>PARTITION_IMPORT_FILE PARTITION_IMPORT_PROMOTE_ASSIGNMENTS  PARTITION_IMPORT_NEW_ASSIGNMENTS PARTITION_IMPORT_EXISTING_ASSIGNMENTS  PARTITION_IMPORT_EXISTING_LOGICLOCK_REGIONS All assignments are exported first automatically, as if you called the "export_assignments" command first, unless the -dont_export_assignments option is specified. You must use the Tcl command "catch" to determine whether the predefined flow ran successfully or not, as in the following example: if {[catch {execute_flow -compile} result]} { puts "\nResult: \$result\n" puts "ERROR: Compilation failed. See report files.\n" } else { puts "\nINFO: Compilation was successful.\n" }</p>		
<b>Example Usage</b>	<pre># To run quartus_map, quartus_fit, quartus_sta, quartus_asm # or other executables based on options. (Refer to "Using # Compilation Flows," "Compiling Designs," and "Specifying # Compiler Settings" in Quartus Prime online Help for more # information.) execute_flow -compile  # To determine if compilation was successful or not # and print out a personalized message. if {[catch {execute_flow -compile} result]} {     puts "\nResult: \$result\n"     puts "ERROR: Compilation failed. See report files.\n" } else {     puts "\nINFO: Compilation was successful.\n"  }  # To perform a full compilation execute_flow -compile</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: Can't run multiple flows simultaneously. Wait for current flow to complete.
TCL_ERROR	1	ERROR: Flow doesn't exist: <string>. Make sure the specified flow exists.
TCL_ERROR	1	ERROR: Only one flow option is allowed. Only one flow can be run for a single command call. If multiple flows are required, use multiple commands.
TCL_ERROR	1	ERROR: Can't find active revision. Make sure there is an open, active revision name. Use the -revision option of project_open, project_new, or use set_current_revision.
TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
TCL_ERROR	1	ERROR: Error(s) found while running an executable. See report file(s) for error message(s). Message log indicates which executable was run last.
TCL_ERROR	1	ERROR: Option -<string> is illegal in the Quartus Prime User Interface. Specify a different option or use a similar command from the Processing menu.
TCL_ERROR	1	ERROR: At least one option is required. Specify at least one option.

### 3.1.11.2. execute\_module (::quartus::flow)

The following table displays information for the execute\_module Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178	
<b>Syntax</b>	execute_module [-h   -help] [-long_help] [-args <arguments>] [-dont_export_assignments] [-tool <asm cdb drc eda fit map syn pow sta stp sim si cpf ipg pfg> ]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-args <arguments>	Option to specify arguments for the executable
	-dont_export_assignments	Option not to export assignments to file. By default, this command exports assignments before running command-line executables.
	-tool <asm cdb drc eda fit map syn pow sta stp sim si cpf ipg pfg>	Option to run the specified executable
<b>Description</b>	Runs one of the command-line executables, such as quartus_map or quartus_fit. If the -args option is specified, the arguments are passed to the command-line executable. All assignments are exported automatically first, as if the "export_assignments" command was called first, unless -dont_export_assignments option is specified. You must use the Tcl command "catch" to determine whether the command-line executable ran successfully or not, as in the following example: if {[catch {execute_module -tool map} result]} { puts "\nResult: \$result\n" } puts "ERROR: Analysis and Synthesis failed. See the report file.\n" else { puts "\nINFO: Analysis and Synthesis was successful.\n" }	
<b>Example Usage</b>	<pre># Run quartus_map using device family Stratix and device part EP1S10B672C6. execute_module -tool map -args "--family=Stratix --part=EP1S10B672C6"  # Compile using a set of executables execute_module -tool map execute_module -tool fit execute_module -tool sta execute_module -tool asm execute_module -tool eda</pre>	

*continued...*

	<pre># To determine if Analysis and Synthesis was successful or not # and print out a personalized message. if {[catch {execute_module -tool map} result]} {     puts "\nResult: \$result\n"     puts "ERROR: Analysis and Synthesis failed. See the report file.\n" } else {     puts "\nINFO: Analysis and Synthesis was successful.\n" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't run multiple flows simultaneously. Wait for current flow to complete.
	TCL_ERROR	1	ERROR: Can't find active revision. Make sure there is an open, active revision name. Use the -revision option of project_open, project_new, or use set_current_revision.
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: Error(s) found while running an executable. See report file(s) for error message(s). Message log indicates which executable was run last.
	TCL_ERROR	1	ERROR: Option is required: -tool. Specify the -tool option.

### 3.1.11.3. get\_flow\_templates (::quartus::flow)

The following table displays information for the get\_flow\_templates Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178		
<b>Syntax</b>	get_flow_templates [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help -long_help		
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.

### 3.1.11.4. write\_flow\_assignment\_digest (::quartus::flow)

The following table displays information for the write\_flow\_assignment\_digest Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178		
<b>Syntax</b>	write_flow_assignment_digest [-h   -help] [-long_help] -flow_name <flow_name>		
<b>Arguments</b>	-h   -help -long_help		
	Long help with examples and possible return values		
<i>continued...</i>			

	-flow_name <flow_name>		The name of the flow that started
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: No revision is currently open. Open a revision.

### 3.1.11.5. write\_flow\_finished (::quartus::flow)

The following table displays information for the write\_flow\_finished Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178		
<b>Syntax</b>	write_flow_finished [-h   -help] [-long_help] [-critical_warnings <critical_warnings>] [-errors <errors>] -flow_name <flow_name> -success <success>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-critical_warnings <critical_warnings>	Set critical warnings message count	
	-errors <errors>	Set error message count	
	-flow_name <flow_name>	The name of the flow that finished	
	-success <success>	Set to 1 if flow finished successfully	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: No revision is currently open. Open a revision.

### 3.1.11.6. write\_flow\_started (::quartus::flow)

The following table displays information for the write\_flow\_started Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178		
<b>Syntax</b>	write_flow_started [-h   -help] [-long_help] -flow_name <flow_name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-flow_name <flow_name>	The name of the flow that started	
<i>continued...</i>			

<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: No revision is currently open. Open a revision.

### 3.1.11.7. `write_flow_template` (::quartus::flow)

The following table displays information for the `write_flow_template` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::flow on page 178		
<b>Syntax</b>	<code>write_flow_template [-h   -help] [-long_help] [-directory &lt;directory&gt; ] -flow_name &lt;flow_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-directory &lt;directory&gt;</code>	Optional name to use as destination for flow template directory	
	<code>-flow_name &lt;flow_name&gt;</code>	The name of the flow template to write	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.

### 3.1.13. ::quartus::insystem\_memory\_edit

The following table displays information for the ::quartus::insystem\_memory\_edit Tcl package:

<b>Tcl Package and Version</b>	::quartus::insystem_memory_edit 1.0
<b>Description</b>	This package contains the set of Tcl functions for reading and editing the contents of memory in an Intel device using the In-System Memory Content Editor.
<b>Availability</b>	This package is loaded by default in the following executables: <code>quartus_stp</code> <code>quartus_stp_tcl</code>
<b>Tcl Commands</b>	<code>begin_memory_edit</code> (::quartus::insystem_memory_edit) on page 184 <code>end_memory_edit</code> (::quartus::insystem_memory_edit) on page 184 <code>get_editable_mem_instances</code> (::quartus::insystem_memory_edit) on page 185 <code>read_content_from_memory</code> (::quartus::insystem_memory_edit) on page 186 <code>save_content_from_memory_to_file</code> (::quartus::insystem_memory_edit) on page 187 <code>update_content_to_memory_from_file</code> (::quartus::insystem_memory_edit) on page 188 <code>write_content_to_memory</code> (::quartus::insystem_memory_edit) on page 189

### 3.1.13.1. begin\_memory\_edit (::quartus::insystem\_memory\_edit)

The following table displays information for the begin\_memory\_edit Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_memory_edit on page 183		
<b>Syntax</b>	begin_memory_edit [-h   -help] [-long_help] -device_name <device name> - hardware_name <hardware name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-device_name <device name>	Name of the device that holds the editable memory instances	
	-hardware_name <hardware name>	Name of the hardware that connects to the JTAG chain	
<b>Description</b>	Start the memory editing sequence. The editing sequence should be terminated with end_memory_edit. The sequence does not have to be terminated unless the device configuration is changed or a different device is edited. The hardware and device name can be obtained with the get.hardware_names and get.device_names commands from the jtag package.		
<b>Example Usage</b>	<pre># Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}  # Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using binary string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "0000001011011100"  # Read back memory content in binary string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2]  # Write memory content using hexadecimal string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "E2F1" - content_in_hex  # Read back memory content in hexadecimal string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 - content_in_hex]  # End the editing sequence end_memory_edit</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: A memory edit sequence has been started. End it first before starting a another one.
	TCL_ERROR	1	ERROR: The specified hardware is not found.

### 3.1.13.2. end\_memory\_edit (::quartus::insystem\_memory\_edit)

The following table displays information for the end\_memory\_edit Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_memory_edit on page 183		
<b>Syntax</b>	end_memory_edit [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	

*continued...*

<b>Description</b>	Terminate the memory editing sequence. The sequence does not have to be terminated unless the device configuration is changed or a different device is edited.									
<b>Example Usage</b>	<pre># Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}  # Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using binary string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "0000001011011100"  # Read back memory content in binary string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2]  # Write memory content using hexadecimal string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "E2F1" -content_in_hex  # Read back memory content in hexadecimal string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 -content_in_hex]  # End the editing sequence end_memory_edit</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: A memory edit sequence has not been started.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: A memory edit sequence has not been started.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: A memory edit sequence has not been started.								

### 3.1.13.3. get\_editable\_mem\_instances (::quartus::insystem\_memory\_edit)

The following table displays information for the `get_editable_mem_instances` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_memory_edit on page 183	
<b>Syntax</b>	<code>get_editable_mem_instances [-h   -help] [-long_help] -device_name &lt;device name&gt; -hardware_name &lt;hardware name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-device_name &lt;device name&gt;</code>	Name of the device that holds the editable memory instances
	<code>-hardware_name &lt;hardware name&gt;</code>	Name of the hardware that connects to the JTAG chain
<b>Description</b>	Retrieve a list of editable memory, ROM, or Ipm_constant instances. A list is returned, each element of which shows the configuration of each instance. This element is another list that specifies the configuration in the following order: <instance index> <depth> <width> <read/write mode> <instance type> <instance name>. The <read/write mode> can be either "RW" or "W"; <instance type> can be either "ROM/RAM" or "CONSTANT". An example showing a list of two instances of different types is shown below: {0 1024 8 RW ROM/RAM mem0} {1 1 32 RW CONSTANT con0} The hardware and device name can be obtained with the <code>get.hardware_names</code> and <code>get.device_names</code> commands from the <code>jtag</code> package. It is recommended that you call this command before the <code>TCL</code> command, <code>begin_memory_edit</code> . Within a memory edit sequence, this command can be applied only to the same device, on which the memory edit sequence has started.	
<b>Example Usage</b>	<pre># List information of all editable memories puts "Information on all editable memories" puts "index,depth,width,mode,type,name" foreach instance [get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"] {     puts "[lindex \$instance 0],[lindex \$instance 1],[lindex \$instance 2],[lindex \$instance 3], [lindex \$instance 4],[lindex \$instance 5]" }</pre>	

*continued...*

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: The TCL command get_editable_mem_instances is called within a memory edit sequence for a different device. End the memory edit first.
	TCL_ERROR	1	ERROR: An internal TCL interpreter error occurred.

### **3.1.13.4. read\_content\_from\_memory (::quartus::insystem\_memory\_edit)**

The following table displays information for the `read_content_from_memory` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::insystem_memory_edit</a> on page 183				
<b>Syntax</b>	<code>read_content_from_memory [-h   -help] [-long_help] [-content_in_hex] -instance_index &lt;instance index&gt; -start_address &lt;starting address&gt; [-timeout &lt;timeout&gt;] -word_count &lt;word count&gt;</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-content_in_hex	The memory content string is represented in hexadecimal format			
	-instance_index <instance index>	Index of the editable memory instance to read			
	-start_address <starting address>	The lowest memory address to be read			
	-timeout <timeout>	amount of time in milliseconds allocated before read times out. Defaults to 10 seconds			
	-word_count <word count>	The number of contiguous memory words to be read			
<b>Description</b>	Retrieves the memory content represented in the bit stream from the specified editable memory instance starting from the specified address. The memory content string is in the same format as the input content string in the TCL command <code>write_content_to_memory</code> .				
<b>Example Usage</b>	<pre># Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0} # Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using binary string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "0000001011011100"  # Read back memory content in binary string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 -timeout 30000]  # Write memory content using hexadecimal string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "E2F1" -content_in_hex  # Read back memory content in hexadecimal string written puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 -content_in_hex]  # End the editing sequence end_memory_edit</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	<i>continued...</i>				

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: A memory edit sequence has not been started.
TCL_ERROR	1	ERROR: The specified word count and the starting address exceeds the specified memory buffer size.
TCL_ERROR	1	ERROR: The specified editable memory instance index is invalid.
TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The device is locked by another application.

### 3.1.13.5. save\_content\_from\_memory\_to\_file (:quartus::insystem\_memory\_edit)

The following table displays information for the save\_content\_from\_memory\_to\_file Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_memory_edit on page 183		
<b>Syntax</b>	save_content_from_memory_to_file [-h   -help] [-long_help] -instance_index <instance index> -mem_file_path <path> -mem_file_type <type> [-timeout <timeout> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-instance_index <instance index>	Index of the editable memory instance to read	
	-mem_file_path <path>	Path to the memory file in which to save the memory content	
	-mem_file_type <type>	Type of the memory file such as "mif" or "hex"	
	-timeout <timeout>	amount of time in milliseconds allocated before read times out. Defaults to 10 seconds	
<b>Description</b>	Retrieves the entire memory contents from the specified editable memory instance starting from address 0 and saves it into the specified memory file.		
<b>Example Usage</b>	<pre># Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using the hex memory file update_content_to_memory_from_file -instance_index 0 -mem_file_path "image_8x1024.hex" - mem_file_type hex  # Read memory content and save back to a hex memory file save_content_from_memory_to_file -instance_index 0 -mem_file_path "exported_image_8x1024.hex" - mem_file_type hex  # Write memory content using the mif memory file update_content_to_memory_from_file -instance_index 0 -mem_file_path "exported_image_8x1024.mif" -mem_file_type mif  # Read memory content and save back to a mif memory file save_content_from_memory_to_file -instance_index 0 -mem_file_path "image_8x1024.mif" - mem_file_type mif -timeout 30000  # End the editing sequence end_memory_edit</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: A memory edit sequence has not been started.
TCL_ERROR	1	ERROR: The specified memory file cannot be written to.
TCL_ERROR	1	ERROR: The specified file type is either invalid or unsupported by this command.
TCL_ERROR	1	ERROR: The specified editable memory instance index is invalid.
TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The device is locked by another application.

### **3.1.13.6. update\_content\_to\_memory\_from\_file (:quartus::insystem\_memory\_edit)**

The following table displays information for the update\_content\_to\_memory\_from\_file Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::insystem_memory_edit</a> on page 183	
<b>Syntax</b>	<pre>update_content_to_memory_from_file [-h   -help] [-long_help] -instance_index &lt;instance index&gt; -mem_file_path &lt;path&gt; -mem_file_type &lt;file type&gt; [-timeout &lt;timeout&gt; ]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-instance_index <instance index>	Index of the editable memory instance to modify
	-mem_file_path <path>	Path to the memory file to load the memory content
	-mem_file_type <file type>	Type of the memory file such as "mif" or "hex"
	-timeout <timeout>	amount of time in milliseconds allocated before write times out. Defaults to 10 seconds
<b>Description</b>	Writes the data stored in the memory file into the specified memory instance starting from address 0.	
<b>Example Usage</b>	<pre># Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using the hex memory file update_content_to_memory_from_file -instance_index 0 -mem_file_path "image_8x1024.hex" -mem_file_type hex  # Read memory content and save back to a hex memory file save_content_from_memory_to_file -instance_index 0 -mem_file_path "exported_image_8x1024.hex" -mem_file_type hex  # Write memory content using the mif memory file update_content_to_memory_from_file -instance_index 0 -mem_file_path "exported_image_8x1024.mif" -mem_file_type mif -timeout 30000  # Read memory content and save back to a mif memory file save_content_from_memory_to_file -instance_index 0 -mem_file_path "image_8x1024.mif" -mem_file_type mif  # End the editing sequence end_memory_edit</pre>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	TCL_ERROR	1
<i>continued...</i>		

TCL_ERROR	1	ERROR: The specified memory file cannot be read because the content is corrupt or the configuration does not match the memory to be updated.
TCL_ERROR	1	ERROR: The specified memory file cannot be opened.
TCL_ERROR	1	ERROR: The specified file type is either invalid or unsupported by this command.
TCL_ERROR	1	ERROR: The specified editable memory instance index is invalid.
TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The device is locked by another application.

### 3.1.13.7. write\_content\_to\_memory (::quartus::insystem\_memory\_edit)

The following table displays information for the `write_content_to_memory` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_memory_edit on page 183	
<b>Syntax</b>	<code>write_content_to_memory [-h   -help] [-long_help] -content &lt;content string&gt; [-content_in_hex] -instance_index &lt;instance index&gt; -start_address &lt;starting address&gt; [-timeout &lt;timeout&gt;] -word_count &lt;word count&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-content &lt;content string&gt;</code>	A string that represents all word values concatenated together in order in either binary or hexadecimal format
	<code>-content_in_hex</code>	The memory content string is represented in hexadecimal format
	<code>-instance_index &lt;instance index&gt;</code>	Index of the editable memory instance to modify
	<code>-start_address &lt;starting address&gt;</code>	The lowest memory address to be modified
	<code>-timeout &lt;timeout&gt;</code>	amount of time in milliseconds allocated before write times out. Defaults to 10 seconds
	<code>-word_count &lt;word count&gt;</code>	The number of contiguous memory words to be modified
<b>Description</b>	Writes the data represented in the bit stream into the specified editable memory instance starting from the specified address. It returns the number of successful writes. The bit stream should be ordered by word from high address to low address, contiguously without gaps or delimiters. If the starting address is ADDR, and word count is N, the order is <word @ ADDR + N - 1> ... <word @ ADDR + 1><word @ ADDR> In each word, the MSB is on the left, LSB is on the right. The bit stream can be in either binary or hexadecimal. For example, if the word width is 8, and two words, 1 and 128, are written to address 0 and 1 respectively, the bitstream should be "1000000000000001" in binary or "8001" in hexadecimal. The TCL command is <code>write_content_to_memory -instance_index 0 -start_address 0 -word_count 2 -content "1000000000000001"</code> or <code>write_content_to_memory -instance_index 0 -start_address 0 -word_count 2 -content "8001" -content_in_hex</code>	
<b>Example Usage</b>	<pre># Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}  # Initiate a editing sequence begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EPI25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"  # Write memory content using binary string write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "0000001011011100" -timeout 30000</pre>	

*continued...*

```

# Read back memory content in binary string written
puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 ]

# Write memory content using hexadecimal string
write_content_to_memory -instance_index 0 -start_address 575 -word_count 2 -content "E2F1" -
content_in_hex

# Read back memory content in hexadecimal string written
puts [read_content_from_memory -instance_index 0 -start_address 575 -word_count 2 -content_in_hex]

# End the editing sequence
end_memory_edit

```

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Data specified in the string does not match the number of bits to update the memory of the specified number of words.
TCL_ERROR	1		ERROR: A memory edit sequence has not been started.
TCL_ERROR	1		ERROR: The specified word count and the starting address exceeds the specified memory buffer size.
TCL_ERROR	1		ERROR: The specified editable memory instance index is invalid.
TCL_ERROR	1		ERROR: The specified editable memory instance index is invalid.
TCL_ERROR	1		ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1		ERROR: The device is locked by another application.

### 3.1.14. ::quartus::insystem\_source\_probe

The following table displays information for the **::quartus::insystem\_source\_probe** Tcl package:

<b>Tcl Package and Version</b>	::quartus::insystem_source_probe 1.0
<b>Description</b>	This package contains the set of Tcl functions for using the In-System Sources and Probes feature to interact with your design in an Intel device.
<b>Availability</b>	This package is loaded by default in the following executables:  quartus_stp quartus_stp_tcl
<b>Tcl Commands</b>	<a href="#">end_insystem_source_probe</a> ( <b>::quartus::insystem_source_probe</b> ) on page 191 <a href="#">get_insystem_source_probe_instance_info</a> ( <b>::quartus::insystem_source_probe</b> ) on page 191 <a href="#">read_probe_data</a> ( <b>::quartus::insystem_source_probe</b> ) on page 192 <a href="#">read_source_data</a> ( <b>::quartus::insystem_source_probe</b> ) on page 193 <a href="#">start_insystem_source_probe</a> ( <b>::quartus::insystem_source_probe</b> ) on page 193 <a href="#">write_source_data</a> ( <b>::quartus::insystem_source_probe</b> ) on page 194

### 3.1.14.1. end\_insystem\_source\_probe (::quartus::insystem\_source\_probe)

The following table displays information for the `end_insystem_source_probe` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190		
<b>Syntax</b>	<code>end_insystem_source_probe [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	This command releases the JTAG chain. Use when finished performing In-System Sources and Probes transactions.		
<b>Example Usage</b>	<pre>#List probe data of instance 0 start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)" puts "probe data of instance 0" puts [read_probe_data -instance_index 0] end_insystem_source_probe</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
	TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.
	TCL_ERROR	1	ERROR: The In-System Sources and Probes instance was not started. This command cannot be used unless the In-System Sources and Probes transaction is started.

### 3.1.14.2. get\_insystem\_source\_probe\_instance\_info (::quartus::insystem\_source\_probe)

The following table displays information for the `get_insystem_source_probe_instance_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190		
<b>Syntax</b>	<code>get_insystem_source_probe_instance_info [-h   -help] [-long_help] -device_name &lt;device name&gt; -hardware_name &lt;hardware name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-device_name <device name>	Name of the device programmed with the design that includes In-System Sources and Probes instances	
	-hardware_name <hardware name>	Name of the hardware that connects to the JTAG chain	
<b>Description</b>	Returns a list of the available In-System Sources and Probes instances and their configuration. {instance_index source_width probe_width instance_name} Example: {0 4 3 src1} {1 5 5 src2} {2 3 6 none}		

*continued...*

<b>Example Usage</b>	<pre># List information of all In-System Sources and Probes instances puts "Information on all In-System Sources and Probes instances" puts "index,source_width,probe_width,name" foreach instance [get_insystem_source_probe_instance_info -hardware_name "USB-Blaster \ [USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"] {     puts "[lindex \$instance 0],[lindex \$instance 1],[lindex \$instance 2],[lindex \$instance 3]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
	TCL_ERROR	1	ERROR: JTAG communication error detected. Errors can be caused by hardware failure or poor signal integrity in the JTAG chain.
	TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.
	TCL_ERROR	1	ERROR: There is already an active In-System Sources and Probes session started. Unable to start another session.
	TCL_ERROR	1	ERROR: The device is locked by another application.

### 3.1.14.3. `read_probe_data` (::quartus::insystem\_source\_probe)

The following table displays information for the `read_probe_data` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190		
<b>Syntax</b>	<code>read_probe_data [-h   -help] [-long_help] -instance_index &lt;instance_index&gt; [-value_in_hex]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-instance_index <instance_index>	Index of the In-System Sources and Probes instance to communicate with	
	-value_in_hex	Specifies that the value string is represented in hexadecimal format	
<b>Description</b>	Retrieves the current value of the probes. A string is returned specifying the status of each probe, with the MSB on the left and LSB on the right. By default, the value is represented as a binary string. Optionally, the option <code>-value_in_hex</code> makes the value a hex string.		
<b>Example Usage</b>	<pre>#List probe data of instance 0 start_insystem_sourc_probe -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)" puts "probe data of instance 0" puts [read_probe_data -instance_index 0] end_insystem_source_probe</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.

*continued...*

TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
TCL_ERROR	1	ERROR: The specified In-System Sources and Probes instance index is invalid.
TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.

### 3.1.14.4. read\_source\_data (::quartus::insystem\_source\_probe)

The following table displays information for the `read_source_data` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190		
<b>Syntax</b>	<code>read_source_data [-h   -help] [-long_help] -instance_index &lt;instance_index&gt; [-value_in_hex]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-instance_index <instance_index>	Index of the In-System Sources and Probes instance to communicate with	
	-value_in_hex	Specifies that the value string is represented in hexadecimal format	
<b>Description</b>	Retrieves the current value of the sources. A string is returned specifying the status of each source, with the MSB on the left and LSB on the right. By default, the value is represented as a binary string. Optionally, the option <code>-value_in_hex</code> makes the value a hex string.		
<b>Example Usage</b>	<pre>#List source data of instance 0 start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)" puts "source data of instance 0" puts [read_source_data -instance_index 0] end_insystem_source_probe</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
	TCL_ERROR	1	ERROR: The specified In-System Sources and Probes instance index is invalid.
	TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.

### 3.1.14.5. start\_insystem\_source\_probe (::quartus::insystem\_source\_probe)

The following table displays information for the `start_insystem_source_probe` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190		
<b>Syntax</b>	<code>start_insystem_source_probe [-h   -help] [-long_help] -device_name &lt;device name&gt; -hardware_name &lt;hardware name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	<i>continued...</i>		

	-long_help	Long help with examples and possible return values	
	-device_name <device name>	Name of the device that holds the In-System Sources and Probes instances	
	-hardware_name <hardware name>	Name of programming hardware connected to the JTAG chain	
<b>Description</b>	Use this command before beginning any In-System Sources and Probes transactions		
<b>Example Usage</b>	<pre>#List probe data of instance 0 start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)" puts "probe data of instance 0" puts [read_probe_data -instance_index 0] end_insystem_source_probe</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
	TCL_ERROR	1	ERROR: JTAG communication error detected. Errors can be caused by hardware failure or poor signal integrity in the JTAG chain.
	TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.
	TCL_ERROR	1	ERROR: There is already an active In-System Sources and Probes session started. Unable to start another session.
	TCL_ERROR	1	ERROR: The device is locked by another application.

### 3.1.14.6. write\_source\_data (::quartus::insystem\_source\_probe)

The following table displays information for the write\_source\_data Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::insystem_source_probe on page 190	
<b>Syntax</b>	write_source_data [-h   -help] [-long_help] -instance_index <instance_index> -value <value> [-value_in_hex]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-instance_index <instance_index>	Index of the In-System Sources and Probes instance to communicate with
	-value <value>	Value for the source
	-value_in_hex	Specify that the value string is represented in hexadecimal format
<b>Description</b>	Sets values for the sources. A value string is sent to the source values. MSB is on the left and LSB is on the right. The value string is truncated on the left (MSB) side if necessary. By default, the values are represented as a binary string. Optionally, the option -value_in_hex makes the values hex strings.	
<b>Example Usage</b>	<pre>#List probe data of instance 0 start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1S25/ _HARDCOPY_FPGA_PROTOTYPE (0x020030DD)" puts "write source data 10010" write_source_data -instance_index 0 -value "10010"</pre>	

*continued...*

	<pre>puts "source data of instance 0" puts [read_source_data -instance_index 0] end_insystem_source_probe</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: An internal Tcl interpreter error occurred.
	TCL_ERROR	1	ERROR: The specified In-System Sources and Probes instance index is invalid.
	TCL_ERROR	1	ERROR: No In-System Sources and Probes instance was found.

### 3.1.15. ::quartus::interactive\_synthesis

The following table displays information for the **::quartus::interactive\_synthesis** Tcl package:

<b>Tcl Package and Version</b>	::quartus::interactive_synthesis 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is loaded by default in the following executable: quartus_syn
<b>Tcl Commands</b>	<a href="#">analyze_files</a> (::quartus::interactive_synthesis) on page 195 <a href="#">check_rtl_connections</a> (::quartus::interactive_synthesis) on page 196 <a href="#">dissolve_rtl_partition</a> (::quartus::interactive_synthesis) on page 196 <a href="#">elaborate</a> (::quartus::interactive_synthesis) on page 197 <a href="#">get_entities</a> (::quartus::interactive_synthesis) on page 197 <a href="#">get_rtl_partition_name</a> (::quartus::interactive_synthesis) on page 197 <a href="#">get_rtl_partitions</a> (::quartus::interactive_synthesis) on page 198 <a href="#">link_rtl_design</a> (::quartus::interactive_synthesis) on page 198 <a href="#">print_ipxact</a> (::quartus::interactive_synthesis) on page 199 <a href="#">report_rtl_assignments</a> (::quartus::interactive_synthesis) on page 199 <a href="#">report_rtl_parameters</a> (::quartus::interactive_synthesis) on page 200 <a href="#">report_rtl_stats</a> (::quartus::interactive_synthesis) on page 200 <a href="#">reset_rtl_design</a> (::quartus::interactive_synthesis) on page 201 <a href="#">save_rtl_design</a> (::quartus::interactive_synthesis) on page 201 <a href="#">synthesize</a> (::quartus::interactive_synthesis) on page 202 <a href="#">uniquify</a> (::quartus::interactive_synthesis) on page 202 <a href="#">write_rtl_report</a> (::quartus::interactive_synthesis) on page 203

#### 3.1.15.1. analyze\_files (::quartus::interactive\_synthesis)

The following table displays information for the **analyze\_files** Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::interactive_synthesis</a> on page 195	
<b>Syntax</b>	<code>analyze_files [-h   -help] [-long_help] [-files &lt;files_value&gt;] [-library &lt;library_value&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-files &lt;files_value&gt;</code>	Specifies the list of files to analyze
	<code>-library &lt;library_value&gt;</code>	Specifies the target library for design units

*continued...*

<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Compile foo.sv into the default library analyze_files -files foo.sv		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.2. check rtl\_connections (::quartus::interactive\_synthesis)

The following table displays information for the `check_rtl_connections` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>check_rtl_connections [-h   -help] [-long_help] [-panel_name &lt;panel_name_value&gt; ]</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-panel_name <panel_name_value>		Specifies the name of the Check Connections report panel
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Report RTL connection issues to "Check RTL Connections" <code>check_rtl_connections -panel_name "Check RTL Connections"</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.15.3. dissolve\_rtl\_partition (::quartus::interactive\_synthesis)

The following table displays information for the `dissolve_rtl_partition` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>dissolve_rtl_partition [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command
	TCL_ERROR	1	ERROR: Partition <string> not found

### 3.1.15.4. elaborate (::quartus::interactive\_synthesis)

The following table displays information for the elaborate Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	elaborate [-h   -help] [-long_help] [-recompile]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-recompile	Enables recompile	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Elaborate from the top-level hierarchy elaborate		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.5. get\_entities (::quartus::interactive\_synthesis)

The following table displays information for the get\_entities Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	get_entities [-h   -help] [-long_help] [-library <library_value>] [-name <name_value>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-library <library_value>	Specifies the library filter	
	-name <name_value>	Specifies the name filter	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Get all "cpu" entities defined in all libraries set cpus [get_entities -entity cpu]		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.6. get\_rtl\_partition\_name (::quartus::interactive\_synthesis)

The following table displays information for the get\_rtl\_partition\_name Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	get_rtl_partition_name [-h   -help] [-long_help] [-name <name_value>]		

*continued...*

<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-name <name_value>	Specifies the name filter
<b>Description</b>	This command currently contains no help description.	
<b>Example Usage</b>	This command currently contains no example usage.	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	TCL_ERROR	1
		INFO: Operation successful
		ERROR: You must open a project before you can use this command

### 3.1.15.7. `get_rtl_partitions` (::quartus::interactive\_synthesis)

The following table displays information for the `get_rtl_partitions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>get_rtl_partitions [-h   -help] [-long_help] [-name &lt;name_value&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name_value>	Specifies the name filter	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Get partitions matching filter pr_region* set pr_regions [get_rtl_partitions -name pr_region*]		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.8. `link_rtl_design` (::quartus::interactive\_synthesis)

The following table displays information for the `link_rtl_design` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>link_rtl_design [-h   -help] [-long_help] [-snapshot &lt;snapshot_value&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-snapshot <snapshot_value>	Specifies input snapshot	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	# Link the RTL design for the current revision link_rtl_design		

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.9. print\_ipxact (::quartus::interactive\_synthesis)

The following table displays information for the `print_ipxact` Tcl command:

Tcl Package and Version	Belongs to ::quartus::interactive_synthesis on page 195		
Syntax	<code>print_ipxact [-h   -help] [-long_help] -print_ipxact_file &lt;file_name&gt;</code>		
Arguments	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-print_ipxact_file <file_name>		Specifies the source file to print
Description	This command currently contains no help description.		
Example Usage	# Generate the IP-XACT for foo.v print_ipxact -print_ipxact_file foo.v		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.10. report rtl\_assignments (::quartus::interactive\_synthesis)

The following table displays information for the `report rtl_assignments` Tcl command:

Tcl Package and Version	Belongs to ::quartus::interactive_synthesis on page 195		
Syntax	<code>report_rtl_assignments [-h   -help] [-long_help] [-instance &lt;instance_name&gt;] [-panel_name &lt;panel_name_value&gt;] [-partition &lt;partition_value&gt;]</code>		
Arguments	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-instance <instance_name>		Specifies the hierarchy path of the instance to report (supports wildcards)
	-panel_name <panel_name_value>		Specifies the name of the report panel
	-partition <partition_value>		Specifies the hierarchy path of the partition(s) to report (supports wildcards)
Description	This command currently contains no help description.		
Example Usage	<code>report_rtl_assignments -panel_name "Source Assignments for Top partition" -partition  </code>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.15.11. report\_rtl\_parameters (::quartus::interactive\_synthesis)

The following table displays information for the `report_rtl_parameters` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::interactive_synthesis</a> on page 195		
<b>Syntax</b>	<code>report_rtl_parameters [-h   -help] [-long_help] [-instance &lt;instance_name&gt;] [-panel_name &lt;panel_name_value&gt;] [-partition &lt;partition_value&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-instance &lt;instance_name&gt;</code>	Specifies the hierarchy path of the instance to report (supports wildcards)	
	<code>-panel_name &lt;panel_name_value&gt;</code>	Specifies the name of the report panel	
	<code>-partition &lt;partition_value&gt;</code>	Specifies the hierarchy path of the partition(s) to report (supports wildcards)	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<code>report_rtl_parameters -panel_name "RTL Parameters for Top partition" -partition  </code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.15.12. report\_rtl\_stats (::quartus::interactive\_synthesis)

The following table displays information for the `report_rtl_stats` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::interactive_synthesis</a> on page 195		
<b>Syntax</b>	<code>report_rtl_stats [-h   -help] [-long_help] [-filename &lt;filename_value&gt;] [-partition &lt;partition_value&gt;] [-stdout]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-filename &lt;filename_value&gt;</code>	Specifies the output filename for the RTL report	
	<code>-partition &lt;partition_value&gt;</code>	Specifies the hierarchy path of the partition to synthesize	
	<code>-stdout</code>	Report to standard output	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.15.13. reset\_rtl\_design (::quartus::interactive\_synthesis)

The following table displays information for the `reset_rtl_design` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>reset_rtl_design [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<pre># Resets the current RTL design. This must be # called prior to calling link_rtl_design again reset_rtl_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.15.14. save\_rtl\_design (::quartus::interactive\_synthesis)

The following table displays information for the `save_rtl_design` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>save_rtl_design [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<pre># Saves the current RTL design for partition Top save_rtl_design -partition    # Saves all partitions in the current RTL design at a particular snapshot # Currently only allow partitioned and synthesized snapshot save_rtl_design -snapshot synthesized</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Snapshot <string> is invalid for save
	TCL_ERROR	1	ERROR: You must open a project before you can use this command
	TCL_ERROR	1	ERROR: Partition <string> not found

### 3.1.15.15. synthesize (::quartus::interactive\_synthesis)

The following table displays information for the `synthesize` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::interactive_synthesis on page 195		
<b>Syntax</b>	<code>synthesize [-h   -help] [-long_help] -partition &lt;partition_value&gt; [-recompile]</code>		
	<i>continued...</i>		

<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-partition <partition_value>	Specifies the hierarchy path of the partition to synthesize
	-recompile	Enables incremental resynthesis algorithm
<b>Description</b>	This command currently contains no help description.	
<b>Example Usage</b>	<pre># Run synthesis on Top partition synthesize -partition   # Run a re-synthesis on Top partition synthesize -partition   -recompile</pre>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
	TCL_ERROR	1
	TCL_ERROR	1
		<b>String Return</b>
		INFO: Operation successful
		ERROR: You must open a project before you can use this command
		ERROR: Partition <string> not found

### 3.1.15.16. uniquify (::quartus::interactive\_synthesis)

The following table displays information for the `uniquify` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::interactive_synthesis</a> on page 195		
<b>Syntax</b>	<code>uniquify [-h   -help] [-long_help] [-analysis_and_elab_report] [-recompile]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-analysis_and_elab_report	Print synthesis reports after analysis and elaboration	
	-recompile	Enables recompile	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<pre># Run uniquify on all partitions as required uniquify  # Run uniquify on one specific partition uniquify -partition cpu_left</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command
	TCL_ERROR	1	ERROR: Current design not found
	TCL_ERROR	1	ERROR: Partition <string> not found

### 3.1.15.17. write\_rtl\_report (::quartus::interactive\_synthesis)

The following table displays information for the `write_rtl_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::interactive_synthesis</a> on page 195		
<b>Syntax</b>	<code>write_rtl_report [-h   -help] [-long_help] [-ascii] [-filename &lt;filename_value&gt;] [-qdb] [-xml]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-ascii	Specifies ASCII as the output format	
	-filename <filename_value>	Specifies the output filename for the RTL report	
	-qdb	Specifies binary as the output format	
	-xml	Specifies XML as the output format	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	<pre># Write the RTL report to the QDB database in binary format write_rtl_report -qdb  # Write the RTL report to the default ASCII report file write_rtl_report -ascii  # Write the RTL report to the XML file write_rtl_report -xml -filename report.xml</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command

### 3.1.16. ::quartus::ipgen

The following table displays information for the `::quartus::ipgen` Tcl package:

<b>Tcl Package and Version</b>	::quartus::ipgen 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is loaded by default in the following executable: <code>quartus_ipgenerate</code>
<b>Tcl Commands</b>	<a href="#">clear_ip_generation_dirs</a> ( <code>::quartus::ipgen</code> ) on page 203 <a href="#">generate_ip_file</a> ( <code>::quartus::ipgen</code> ) on page 204 <a href="#">generate_project_ip_files</a> ( <code>::quartus::ipgen</code> ) on page 205 <a href="#">get_project_ip_files</a> ( <code>::quartus::ipgen</code> ) on page 206

#### 3.1.16.1. clear\_ip\_generation\_dirs (::quartus::ipgen)

The following table displays information for the `clear_ip_generation_dirs` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::ipgen</a> on page 203
<b>Syntax</b>	<code>clear_ip_generation_dirs [-h   -help] [-long_help]</code>

*continued...*

<b>Arguments</b>	-h   -help -long_help		
	Long help with examples and possible return values		
<b>Description</b>	This command removes all the generation directories for the Platform Designer IP files in the opened project. All the content in the generation directories will be removed.		
<b>Example Usage</b>	# Remove all the generation directories for the Platform Designer IP files in the opened project project_open my_project clear_ip_generation_dirs		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: The command failed with an unknown error.

### 3.1.16.2. generate\_ip\_file (::quartus::ipgen)

The following table displays information for the generate\_ip\_file Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::ipgen on page 203	
<b>Syntax</b>	generate_ip_file [-h   -help] [-long_help] [-clean] [-clear_ip_generation_dirs] [-simulation <verilog vhdl>] [-simulator <modelsim vcs vcsmx riviera xcelium>] [-synthesis <verilog vhdl>] <file>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-clean	Specify whether pre-existing generation directories should be cleared before generation.
	-clear_ip_generation_dirs	Specify whether pre-existing generation directories should be cleared before generation.
	-simulation <verilog vhdl>	Set the simulation target type. Valid values are verilog or vhdl.
	-simulator <modelsim vcs vcsmx riviera xcelium>	Set the simulator target type. Valid values are modelsim, vcs, vcsmx, riviera, and/or xcelium.
	-synthesis <verilog vhdl>	Set the synthesis target type. Valid values are verilog or vhdl.
	<file>	A Platform Designer IP file path. -file="path1;path2"
<b>Description</b>	This command generates the files for a specified Platform Designer IP in the opened project. --synthesis <value>: Specify the synthesis target type. Valid values are verilog or vhdl. This is not a required option. When not specified, it defaults to verilog. --simulation <value>: Specify the simulation target type. Valid values are verilog or vhdl. This is not a required option. When not specified, no simulation files are generated. --simulator <value>: Specify the simulator target type. Valid values are modelsim, vcs, vcsmx, riviera, xcelium. This is not a required option. When not specified, simulation files for all simulators are generated. --clear_ip_generation_dirs: Specify whether pre-existing generation directories should be cleared before generation. This is not a required option. When not specified, the generation directories will not be cleared. --clean: Specify whether pre-existing generation directories should be cleared before generation. This option is a short version of the clear_ip_generation_dirs option. This is not a required option. When not specified, the generation directories will not be cleared.	
<b>Example Usage</b>	# generate the specified Platform Designer IP in the project with the specified targets. Clear any pre-existing # generation directories before performing the generation.	

*continued...*

	<pre>project_open my_project generate_ip_file my_ip_file.qsys -synthesis verilog -simulation verilog -simulator modelsim - clear_ip_generation_dirs</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The file <string> does not exist in project.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: The command failed with an unknown error.

### 3.1.16.3. generate\_project\_ip\_files (::quartus::ipgen)

The following table displays information for the generate\_project\_ip\_files Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::ipgen on page 203	
<b>Syntax</b>	<pre>generate_project_ip_files [-h   -help] [-long_help] [-clean] [-clear_ip_generation_dirs] [-parallel &lt;on off&gt;] [-simulation &lt;verilog vhdl&gt;] [-simulator &lt;modelsim vcs vcsmx riviera xcelium&gt;] [-synthesis &lt;verilog vhdl&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-clean	Specify whether pre-existing generation directories should be cleared before generation.
	-clear_ip_generation_dirs	Specify whether pre-existing generation directories should be cleared before generation.
	-parallel <on off>	Specify whether to allow parallel IP generation. Specify off to disable parallel
	-simulation <verilog vhdl>	Set the simulation target type. Valid values are verilog or vhdl.
	-simulator <modelsim vcs vcsmx riviera xcelium>	Set the simulator target type. Valid values are modelsim, vcs, vcsmx, riviera, and/or xcelium.
	-synthesis <verilog vhdl>	Set the synthesis target type. Valid values are verilog or vhdl.
<b>Description</b>	<p>This command generates the files for all Platform Designer IP in the opened project. If no option is specified, this command generates the verilog synthesis target only. --synthesis &lt;value&gt;: Specify the synthesis target type. Valid values are verilog or vhdl. This is not a required option. When not specified, it defaults to verilog. --simulation &lt;value&gt;: Specify the simulation target type. Valid values are verilog or vhdl. This is not a required option. When not specified, no simulation files are generated. --simulator &lt;value&gt;: Specify the simulator target type. Valid values are modelsim, vcs, vcsmx, riviera, and/or xcelium. This is not a required option. When not specified, simulation files for all simulators are generated. --clear_ip_generation_dirs: Specify whether pre-existing generation directories should be cleared before generation. This is not a required option. When not specified, the generation directories will not be cleared. --clean: Specify whether pre-existing generation directories should be cleared before generation. This option is a short version of the clear_ip_generation_dirs option. This is not a required option. When not specified, the generation directories will not be cleared. --parallel &lt;on,off&gt;: Specify whether to allow parallel IP generation. When this is on, parallel processing will be enable if project or global Quartus setting is enabled. When this setting is off, parallel is disabled regardless of project or global Quartus settings.</p>	
<b>Example Usage</b>	<pre># generate all the Platform Designer IP in the project with the specified targets. Clear any pre-existing # generation directories before performing the generation.</pre>	

*continued...*

	<pre>project_open my_project generate_project_ip_files -synthesis verilog -simulation verilog -simulator modelsim - clear_ip_generation_dirs</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The file <string> does not exist in project.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: The command failed with an unknown error.

### 3.1.16.4. get\_project\_ip\_files (::quartus::ipgen)

The following table displays information for the `get_project_ip_files` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::ipgen</a> on page 203		
<b>Syntax</b>	<code>get_project_ip_files [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	This command returns a Tcl list containing the full path of the Platform Designer IP files (.qsys or .ip) found in the opened project.		
<b>Example Usage</b>	<pre>project_open my_project get_project_ip_files</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: The command failed with an unknown error.

### 3.1.17. ::quartus::iptclgen

The following table displays information for the `::quartus::iptclgen` Tcl package:

<b>Tcl Package and Version</b>	::quartus::iptclgen 1.0
<b>Description</b>	This package contains the set of Tcl functions for generating Memory IP.
<b>Availability</b>	This package is available for loading in the following executables:  <code>qpro</code> <code>qpro_sh</code> <code>quartus</code> <code>quartus_cdb</code> <code>quartus_sh</code>
<b>Tcl Commands</b>	<a href="#">compute_pll</a> ( <code>::quartus::iptclgen</code> ) on page 207 <a href="#">generate_vhdl_simgen_model</a> ( <code>::quartus::iptclgen</code> ) on page 207 <a href="#">parse_hdl</a> ( <code>::quartus::iptclgen</code> ) on page 208 <a href="#">parse_tcl</a> ( <code>::quartus::iptclgen</code> ) on page 209

### 3.1.17.1. compute\_pll (::quartus::iptclgen)

The following table displays information for the `compute_pll` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::iptclgen</code> on page 206		
<b>Syntax</b>	<code>compute_pll [-h   -help] [-long_help] -family &lt;family&gt; -input_freq &lt;input_freq&gt; -output_freqs &lt;output_freqs&gt; -output_phases &lt;output_phases&gt; -pll_type &lt;pll_type&gt; -speed_grade &lt;speed_grade&gt; -temp_dir &lt;temp_dir&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-family &lt;family&gt;</code>	Device family	
	<code>-input_freq &lt;input_freq&gt;</code>	input frequency in Mhz	
	<code>-output_freqs &lt;output_freqs&gt;</code>	desired output frequencies	
	<code>-output_phases &lt;output_phases&gt;</code>	desired output phase shifts	
	<code>-pll_type &lt;pll_type&gt;</code>	pll type	
	<code>-speed_grade &lt;speed_grade&gt;</code>	Device speed grade	
	<code>-temp_dir &lt;temp_dir&gt;</code>	temporary directory	
<b>Description</b>	Parses the HDL file specified and saves output to the file name specified.		
<b>Example Usage</b>	<code>compute_pll -family "stratix iii" -pll_type "fast_pll" -input_freq 100 -output_freqs "100, 200, 333"</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid Intel FPGA IP Name: <string>. Specify a legal port type.
	TCL_ERROR	1	ERROR: Illegal number of arguments. Specify %u argument(s) for option <string>.
	TCL_ERROR	1	ERROR: No open project. Open an existing project or create a new project.

### 3.1.17.2. generate\_vhdl\_simgen\_model (::quartus::iptclgen)

The following table displays information for the `generate_vhdl_simgen_model` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::iptclgen</code> on page 206		
<b>Syntax</b>	<code>generate_vhdl_simgen_model [-h   -help] [-long_help] -blackbox &lt;blackbox&gt; -family &lt;family&gt; -files &lt;files&gt; -result_dir &lt;result_dir&gt; -temp_dir &lt;temp_dir&gt; -top_level_name &lt;top_level_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-blackbox &lt;blackbox&gt;</code>	comma-separated list of modules that should be blackboxed	
	<code>-family &lt;family&gt;</code>	family name	

*continued...*

	-files <files>	comma seperated list of files	
	-result_dir <result_dir>	result directory for .vho file. Must already exist	
	-temp_dir <temp_dir>	temp_dir	
	-top_level_name <top_level_name>	top level entity name	
<b>Description</b>	Creates a temporary project in the temporary directory, creates the simgen model and copies the model to result_dir.		
<b>Example Usage</b>	<pre>generate_vhdl_simgen_model -family "stratix iii" -files "mycore.v,subcore.v" -top_level_name "mycore.v" -temp_dir "c:/temp" -result_dir "c:/outdir" -blackbox "blackboxme"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No open project. Open an existing project or create a new project.

### 3.1.17.3. parse\_hdl (::quartus::iptclgen)

The following table displays information for the parse\_hdl Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::iptclgen</a> on page 206		
<b>Syntax</b>	<pre>parse_hdl [-h   -help] [-long_help] -core_params &lt;core_params&gt; -indir_name &lt;indir_name&gt; -inhd़l_files &lt;inhd़l_files&gt; [-module_list &lt;module_list&gt;] -module_name &lt;module_name&gt; -outdir_name &lt;outdir_name&gt; -supported_params &lt;supported_params&gt;</pre>		
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help Short help</li> <li>-long_help Long help with examples and possible return values</li> <li>-core_params &lt;core_params&gt; Comma-delimited list of core parameters</li> <li>-indir_name &lt;indir_name&gt; input directory of hdl files</li> <li>-inhd़l_files &lt;inhd़l_files&gt; input hdl name</li> <li>-module_list &lt;module_list&gt; list of modules to uniquify name</li> <li>-module_name &lt;module_name&gt; toplevel module name</li> <li>-outdir_name &lt;outdir_name&gt; output directory of hdl files</li> <li>-supported_params &lt;supported_params&gt; Comma-delimited list of supported core parameters</li> </ul>		
<b>Description</b>	Parses the HDL file specified and saves output to the file name specified.		
<b>Example Usage</b>	<pre>parse_hdl -inhd़l_files rldram_dev.v -core_params "USE_CLK, STRATIXIII" -outdir_name "/project/hdl"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid Intel FPGA IP Name: <string>. Specify a legal port type.
	TCL_ERROR	1	ERROR: Illegal number of arguments. Specify %u argument(s) for option <string>.
	TCL_ERROR	1	ERROR: No open project. Open an existing project or create a new project.

### 3.1.17.4. parse\_tcl (::quartus::iptclgen)

The following table displays information for the `parse_tcl` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::iptclgen</code> on page 206		
<b>Syntax</b>	<code>parse_tcl [-h   -help] [-long_help] [-core_params &lt;core_params&gt;] [-core_sub_params &lt;core_sub_params&gt;] -indir_name &lt;indir_name&gt; -intcl_files &lt;intcl_files&gt; -module_name &lt;module_name&gt; -outdir_name &lt;outdir_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-core_params &lt;core_params&gt;</code>	Comma-delimited list of core parameters	
	<code>-core_sub_params &lt;core_sub_params&gt;</code>	map of parameter-to-value pairs to replace	
	<code>-indir_name &lt;indir_name&gt;</code>	input directory of tcl files	
	<code>-intcl_files &lt;intcl_files&gt;</code>	input tcl name	
	<code>-module_name &lt;module_name&gt;</code>	toplevel module name	
	<code>-outdir_name &lt;outdir_name&gt;</code>	output directory of tcl files	
<b>Description</b>	Parses the HDL file specified and saves output to the file name specified.		
<b>Example Usage</b>	<code>parse_tcl -intcl_files rldram_dev.v -core_params "USE_CLK, STRATIXIII" -outdir_name "/project/tcl"</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Invalid Intel FPGA IP Name: <string>. Specify a legal port type.
	TCL_ERROR	1	ERROR: Illegal number of arguments. Specify %u argument(s) for option <string>.
	TCL_ERROR	1	ERROR: No open project. Open an existing project or create a new project.

### 3.1.18. ::quartus::jtag

The following table displays information for the `::quartus::jtag` Tcl package:

<b>Tcl Package and Version</b>	::quartus::jtag 1.0
<b>Description</b>	This package contains the set of Tcl functions for controlling the JTAG chain using Intel programming hardware.
<b>Availability</b>	This package is loaded by default in the following executables: <code>quartus_stp</code> <code>quartus_stp_tcl</code>
<b>Tcl Commands</b>	<code>close_device</code> ( <code>::quartus::jtag</code> ) on page 210 <code>device_dr_shift</code> ( <code>::quartus::jtag</code> ) on page 210 <code>device_ir_shift</code> ( <code>::quartus::jtag</code> ) on page 212 <code>device_lock</code> ( <code>::quartus::jtag</code> ) on page 213 <code>device_run_test_idle</code> ( <code>::quartus::jtag</code> ) on page 214 <code>device_unlock</code> ( <code>::quartus::jtag</code> ) on page 215 <code>device_virtual_dr_shift</code> ( <code>::quartus::jtag</code> ) on page 216 <code>device_virtual_ir_shift</code> ( <code>::quartus::jtag</code> ) on page 217

*continued...*

get_device_names (::quartus::jtag) on page 219 get_hardware_names (::quartus::jtag) on page 220 open_device (::quartus::jtag) on page 220
---

### **3.1.18.1. close\_device (::quartus::jtag)**

The following table displays information for the `close_device` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::jtag</a> on page 209		
<b>Syntax</b>	<code>close_device [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	End the shared communication with the opened device.		
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get_hardware_names] {   puts \$hardware_name   if { [string match "USB-Blaster*" \$hardware_name] } {     set usbblaster_name \$hardware_name   } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {   puts \$device_name   if { [string match "@1*" \$device_name] } {     set test_device \$device_name   } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No device has been opened.
	TCL_ERROR	1	ERROR: A device was locked.

### **3.1.18.2. device\_dr\_shift (::quartus::jtag)**

The following table displays information for the `device_dr_shift` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::jtag</a> on page 209		
<b>Syntax</b>	<code>device_dr_shift [-h   -help] [-long_help] [-dr_value &lt;data register value&gt;] - length &lt;data register length&gt; [-no_captured_dr_value] [-value_in_hex]</code>		
	<i>continued...</i>		

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-dr_value &lt;data register value&gt;</code>	Value of string operand type in either default binary or hexadecimal format to be written into the data register in the JTAG tap controller of the open device	
	<code>-length &lt;data register length&gt;</code>	Length of the data register in the JTAG tap controller in the open device	
	<code>-no_captured_dr_value</code>	Option not to return the data instruction register value. If this is specified, this DR scan may be packed together with the subsequent IR or DR scan until the device is unlocked or a captured value is requested	
	<code>-value_in_hex</code>	Option to specify that the value string is represented in hexadecimal format	
<b>Description</b>	Writes the specified value into the data register of the JTAG tap controller of the open device. Returns the captured data register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together. The value is specified using either a binary string or a hexadecimal string. The bit on the left most side is the first bit shifted in. For example, using binary string "010001", the first bit shifted into the dr register is 1; the last bit is 0. The same string can be represented in hexadecimal as "11". The device must be locked before you can perform this operation.		
<b>Example Usage</b>	<pre># List all available programming hardware, and select the USB-Blaster. # (Note: this example assumes only one USB-Blaster is connected.) puts "Programming Hardware:" foreach hardware_name [get.hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbbblaster_name.\n"  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get.device_names -hardware_name \$usbbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n"  # Open device open_device -hardware_name \$usbbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Captured value cannot be disabled at the time when no value is shifted into data register.

*continued...*

TCL_ERROR	1	ERROR: A device has not been locked; exclusive communication must be obtained first.
TCL_ERROR	1	ERROR: A device has been locked by another application; exclusive communication cannot be granted within the specified timeout period.
TCL_ERROR	1	ERROR: The length of the value string specified does not match the length parameter specified.

### 3.1.18.3. device\_ir\_shift (::quartus::jtag)

The following table displays information for the device\_ir\_shift Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209	
<b>Syntax</b>	device_ir_shift [-h   -help] [-long_help] -ir_value <instruction register value> [-no_captured_ir_value]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-ir_value <instruction register value>	Value to be written into the instruction register in the JTAG tap controller of the open device. Operand must be of a TCL numerical type such as decimal (10), hexadecimal (0xa), or octal number (012)
	-no_captured_ir_value	Option to not return the captured instruction register value. If this is specified, this IR scan may be packed together with the subsequent IR or DR scan until the device is unlocked or a captured value is requested
<b>Description</b>	Writes the specified value into the instruction register of the JTAG tap controller of the open device. Returns the captured instruction register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together. The instruction register length is determined automatically by the Quartus Prime JTAG server. The device must be locked first before this operation.	
<b>Example Usage</b>	<pre># List all available programming hardware, and select the USB-Blaster. # (Note: this example assumes only one USB-Blaster is connected.) puts "Programming Hardware:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "01*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock</pre>	

*continued...*

	# Close device close_device		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: A device has not been locked; exclusive communication must be obtained first.
	TCL_ERROR	1	ERROR: A device has been locked by another application; exclusive communication cannot be granted within the specified timeout period.

### 3.1.18.4. device\_lock (::quartus::jtag)

The following table displays information for the device\_lock Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209				
<b>Syntax</b>	device_lock [-h   -help] [-long_help] -timeout <timeout>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-timeout <timeout>	The amount of time in millisecond to wait for the access to the device.			
<b>Description</b>	Obtain an exclusive JTAG communication to the device for the subsequent IR and DR shift operations. The device must be locked before any instruction and/or data register shift operation. This should be used as little time as possible as it denies the access of other applications to this chain. The command, unlock, should be called as soon as possible to allow other applications to access the device.				
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get.hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n"  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get.device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n"  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: No device has been opened.
TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure and signal integrity in the JTAG chain. Try to restart.
TCL_ERROR	1	ERROR: A device was locked.
TCL_ERROR	1	ERROR: A device has been locked by another application; exclusive communication cannot be granted within the specified timeout period.

### 3.1.18.5. `device_run_test_idle` (::quartus::jtag)

The following table displays information for the `device_run_test_idle` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209				
<b>Syntax</b>	<code>device_run_test_idle [-h   -help] [-long_help] [-num_clocks &lt;state cycle count&gt; ]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-num_clocks &lt;state cycle count&gt;</code>	The number of times the Run_Test_Idle state is cycled through. If not specified, this value is 1			
<b>Description</b>	Drive the JTAG controller into the Run_Test_Idle state for a number cycles specified with the -num_clocks option. The device must be locked before you can perform this operation.				
<b>Example Usage</b>	<pre># List all available programming hardware, and select the USB-Blaster. # (Note: this example assumes only one USB-Blaster is connected.) puts "Programming Hardware:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: Ox[device_dr_shift -length 32 -value_in_hex]"  # Goto the Run_Test_Idle state and stay there for 8 cycles. device_run_test_idle -num_clocks 8 device_unlock  # Close device close_device</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: A device has not been locked; exclusive communication must be obtained first.
TCL_ERROR	1	ERROR: A device has been locked by another application; exclusive communication cannot be granted within the specified timeout period.

### 3.1.18.6. device\_unlock (::quartus::jtag)

The following table displays information for the device\_unlock Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209		
<b>Syntax</b>	device_unlock [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Release the exclusive JTAG communication lock on the device.		
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get.hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n"  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get.device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No device has been opened.
	TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure and signal integrity in the JTAG chain. Try to restart.
	TCL_ERROR	1	ERROR: A device has not been locked; exclusive communication must be obtained first.

### 3.1.18.7. device\_virtual\_dr\_shift (::quartus::jtag)

The following table displays information for the device\_virtual\_dr\_shift Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209	
<b>Syntax</b>	device_virtual_dr_shift [-h   -help] [-long_help] [-dr_value <data register value>] -instance_index <instance index> -length <data register length> [-no_captured_dr_value] [-show_equivalent_device_ir_dr_shift] [-value_in_hex]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-dr_value <data register value>	Value of string operand type in either default binary or hexadecimal format to be written into the data register in this instance
	-instance_index <instance index>	The index of the virtual JTAG instance
	-length <data register length>	Length of the data register in this instance
	-no_captured_dr_value	Option to not return the data instruction register value
	-show_equivalent_device_ir_dr_shift	Option to show equivalent device ir dr shifts performed by this command
	-value_in_hex	Option to specify that the value string is represented in hexadecimal format
<b>Description</b>	Writes the specified value into the data register of the JTAG tap controller of the open device. Returns the captured data register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together. The value is specified using either a binary string or a hexadecimal string. The bit on the left most side is the first bit shifted in. For example, using the binary string "010001", the first bit shifted into the dr register is 1; the last bit is 0. The same string can be represented in hexadecimal as "11". The device must be locked first, and the target instance must be activated using the device_virtual_ir_shift command before this operation. Moreover, the device should be locked before the virtual IR shift operation to prevent another application from activating another instance.	
<b>Example Usage</b>	<pre># List all available programming hardware, and select the USB-Blaster. # (Note: this example assumes only one USB-Blaster is connected.) puts "Programming Hardware:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # The follow virtual JTAG IR and DR shift sequence engage with # the example virtual JTAG instance. # # Two instructions: SAMPLE (1) FEED (2) # SAMPLE instruction samples a 8-bit bus; the captured value shows the number of sample # performed. # FEED instruction supplies a 8-bit value to the logic connected to this instance. # Both data registers corresponding to the IR are 8 bit wide.</pre>	

*continued...*

```

# Send SAMPLE instruction to IR, read captured IR for the sampling number.
# Capture the DR register for the current sampled value.
device_lock -timeout 10000
puts "Current LED Value (sample #[device_virtual_ir_shift -instance_index 0 -ir_value 1]):"
[device_virtual_dr_shift -instance_index 0 -length 8 -value_in_hex]"
device_unlock

# Send FEED instruction to IR, read a two-digit hex string from the console,
# then send the new value to the DR register.
puts "\nType in 2 digits in hexadecimal to update the LED:"
gets stdin update_value

device_lock -timeout 10000
device_virtual_ir_shift -instance_index 0 -ir_value 2 -no_captured_ir_value
device_virtual_dr_shift -instance_index 0 -length 8 -dr_value $update_value -value_in_hex -
no_captured_dr_value
device_unlock

# Close device
close_device

```

Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Captured value cannot be disabled at the time when no value is shifted into data register.
TCL_ERROR	1		ERROR: The specified virtual JTAG instance cannot be found.
TCL_ERROR	1		ERROR: One of the options, mfg_id, type_id and version is specified, but not all. All of them are required.
TCL_ERROR	1		ERROR: A device has not been locked; exclusive communication must be obtained first.
TCL_ERROR	1		ERROR: A device has been locked by another application; exclusive communication cannot be granted within the specified timeout period.
TCL_ERROR	1		ERROR: The length of the value string specified does not match the length parameter specified.

### 3.1.18.8. device\_virtual\_ir\_shift (::quartus::jtag)

The following table displays information for the device\_virtual\_ir\_shift Tcl command:

Tcl Package and Version	Belongs to ::quartus::jtag on page 209	
Syntax	device_virtual_ir_shift [-h   -help] [-long_help] -instance_index <instance index> -ir_value <instruction register value> [-no_captured_ir_value] [-show_equivalent_device_ir_dr_shift]	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-instance_index <instance index>	The index of the virtual JTAG instance
	-ir_value <instruction register value>	Value to be written into the instruction register in this instance. Operand must be of a TCL numerical type such as decimal (10), hexadecimal (0xa), or octal number (012)
	-no_captured_ir_value	Option to not return the captured instruction register value. If this is specified, this IR scan may be packed together with the subsequent IR or DR scan until the device is unlocked or a captured value is requested

*continued...*

	<pre>- show_equivalent_device_ir_dr_shift</pre>	Option to show equivalent device ir and dr shifts performed by this command	
<b>Description</b>	Writes the specified value into the instruction register of the specified virtual JTAG instance in the open device. Returns the captured instruction register value. You can disable the captured value return to improve the JTAG communication speed by packing multiple IR or DR scans together. The command also activates the target instance such that the consequent virtual DR shift operations are applied to this instance before the device is unlocked. Before any virtual DR shift operation, this command must be executed first to activate the instance. The device must be locked first before this operation.		
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # The follow virtual JTAG IR and DR shift sequence engage with # the example virtual JTAG instance. # # Two instructions: SAMPLE (1) FEED (2) # SAMPLE instruction samples a 8-bit bus; the captured value shows the number of sample # performed. # FEED instruction supplies a 8-bit value to the logic connected to this instance. # Both data registers corresponding to the IR are 8 bit wide.  # Send SAMPLE instruction to IR, read captured IR for the sampling number. # Capture the DR register for the current sampled value. device_lock -timeout 10000 puts "Current LED Value (sample #[device_virtual_ir_shift -instance_index 0 -ir_value 1]):" [device_virtual_dr_shift -instance_index 0 -length 8 -value_in_hex]" device_unlock  # Send FEED instruction to IR, read a two-digit hex string from the console, # then send the new value to the DR register. puts "\nType in 2 digits in hexadecimal to update the LED:" gets stdin update_value device_lock -timeout 10000 device_virtual_ir_shift -instance_index 0 -ir_value 2 -no_captured_ir_value device_virtual_dr_shift -instance_index 0 -length 8 -dr_value \$update_value -value_in_hex - no_captured_dr_value device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The specified virtual JTAG instance cannot be found.
	TCL_ERROR	1	ERROR: One of the options, mfg_id, type_id and version is specified, but not all. All of them are required.
	TCL_ERROR	1	ERROR: A device has not been locked; exclusive communication must be obtained first.

### 3.1.18.9. get\_device\_names (::quartus::jtag)

The following table displays information for the `get_device_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::jtag on page 209		
<b>Syntax</b>	<code>get_device_names [-h   -help] [-long_help] -hardware_name &lt;hardware name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-hardware_name &lt;hardware name&gt;</code>	The name of the programming hardware that connects to the JTAG chain. The name can be obtained from command: <code>get_hardware_names</code> .	
<b>Description</b>	Retrieves a list of names of the devices on the JTAG chain to which the specified programming hardware is connected. The name of the device is in the following format: <number on circuit board>: <JTAG ID code>: <device name>. For example, in the device name @1: 0x082000DD: EP20K200C, @1 indicates that it is the first device on the circuit board, 0x082000DD is the JTAG ID code for the device, and EP20K200C is the device name.		
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No device is detected in the specified JTAG chain.
	TCL_ERROR	1	ERROR: The specified hardware is not found.

### 3.1.18.10. get\_hardware\_names (::quartus::jtag)

The following table displays information for the `get_hardware_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::jtag</a> on page 209		
<b>Syntax</b>	<code>get_hardware_names [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Retrieves a list of the names of the programming hardware attached to and configured for the JTAG server. The hardware name is in the following format: <hardware type> [<port>]. For example, in the hardware name USB-Blaster [USB-0], USB-Blaster is the name of the programming hardware, and USB-0 is the name of the port to which the hardware is connected.		
<b>Example Usage</b>	<pre># List all available programming hardware, and select the USB-Blaster. # (Note: this example assumes only one USB-Blaster is connected.) puts "Programming Hardware:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n";  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n";  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No programming hardware is attached to the JTAG server or it is not configured properly.

### 3.1.18.11. open\_device (::quartus::jtag)

The following table displays information for the `open_device` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::jtag</a> on page 209		
<b>Syntax</b>	<code>open_device [-h   -help] [-long_help] -device_name &lt;device name&gt; -hardware_name &lt;hardware name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
<i>continued...</i>			

	-long_help	Long help with examples and possible return values	
	-device_name <device name>	The name of the device on the JTAG chain. The name can be obtained from command: get_device_names	
	-hardware_name <hardware name>	The name of the programming hardware that connects to the JTAG chain. The name can be obtained from command: get_hardware_names	
<b>Description</b>	Initiate a shared JTAG communication with the device. The command, close_device, is called to end the communication with the device. Only one device can be opened per process. Multiple devices can be opened in multiple processes.		
<b>Example Usage</b>	<pre># List all available programming hardwares, and select the USBBlaster. # (Note: this example assumes only one USBBlaster connected.) puts "Programming Hardwares:" foreach hardware_name [get_hardware_names] {     puts \$hardware_name     if { [string match "USB-Blaster*" \$hardware_name] } {         set usbblaster_name \$hardware_name     } } puts "\nSelect JTAG chain connected to \$usbblaster_name.\n"  # List all devices on the chain, and select the first device on the chain. puts "\nDevices on the JTAG chain:" foreach device_name [get_device_names -hardware_name \$usbblaster_name] {     puts \$device_name     if { [string match "@1*" \$device_name] } {         set test_device \$device_name     } } puts "\nSelect device: \$test_device.\n"  # Open device open_device -hardware_name \$usbblaster_name -device_name \$test_device  # Retrieve device id code. # IDCODE instruction value is 6; The ID code is 32 bits long.  # IR and DR shift should be locked together to ensure that other applications # will not change the instruction register before the id code value is shifted # out while the instruction register is still holding the IDCODE instruction. device_lock -timeout 10000 device_ir_shift -ir_value 6 -no_captured_ir_value puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]" device_unlock  # Close device close_device</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: A device was opened. Only one device can be open at a time within this process. Close previously opened device first.
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.

### 3.1.19. ::quartus::logic\_analyzer\_interface

The following table displays information for the **::quartus::logic\_analyzer\_interface** Tcl package:

<b>Tcl Package and Version</b>	::quartus::logic_analyzer_interface 1.0
<b>Description</b>	This package contains the set of Tcl functions for querying and modifying the Logic Analyzer Interface output pin state in an Intel device.
<i>continued...</i>	

<b>Availability</b>	This package is loaded by default in the following executables:  quartus_stp quartus_stp_tcl
<b>Tcl Commands</b>	<a href="#">begin_logic_analyzer_interface_control</a> (::quartus::logic_analyzer_interface) on page 222 <a href="#">change_bank_to_output_pin</a> (::quartus::logic_analyzer_interface) on page 223 <a href="#">end_logic_analyzer_interface_control</a> (::quartus::logic_analyzer_interface) on page 224 <a href="#">get_current_state_of_output_pin</a> (::quartus::logic_analyzer_interface) on page 225 <a href="#">tristate_output_pin</a> (::quartus::logic_analyzer_interface) on page 226

### 3.1.19.1. begin\_logic\_analyzer\_interface\_control (::quartus::logic\_analyzer\_interface)

The following table displays information for the begin\_logic\_analyzer\_interface\_control Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::logic_analyzer_interface</a> on page 221				
<b>Syntax</b>	<code>begin_logic_analyzer_interface_control [-h   -help] [-long_help] -device_name &lt;device name&gt; -file_path &lt;file path&gt; -hardware_name &lt;hardware name&gt;</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-device_name <device name>	Name of the device to be controlled			
	-file_path <file path>	Path of the Logic Analyzer Interface (.lai) file			
	-hardware_name <hardware name>	Name of the hardware that connects to the JTAG chain			
<b>Description</b>	Starts the Logic Analyzer Interface control sequence to query the Logic Analyzer Interface output pin state and change output pins state. The control sequence should be terminated with end_logic_analyzer_interface_control. The hardware and device name can be obtained by using get.hardware_names and get.device_names respectively from the jtag package.				
<b>Example Usage</b>	<pre># Start a new control sequence. begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path "lai_demo.lai"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Tristate the output pins tristate_output_pin -instance_name "auto_lai_0"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # End the control sequence. end_logic_analyzer_interface_control</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: A Logic Analyzer Interface control sequence has been started already.
TCL_ERROR	1	ERROR: The specified device is not found.
TCL_ERROR	1	ERROR: The Logic Analyzer Interface file (.lai) cannot be opened, or it is an invalid file.
TCL_ERROR	1	ERROR: The specified hardware is not found.

### 3.1.19.2. change\_bank\_to\_output\_pin (::quartus::logic\_analyzer\_interface)

The following table displays information for the `change_bank_to_output_pin` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::logic_analyzer_interface on page 221	
<b>Syntax</b>	<code>change_bank_to_output_pin [-h   -help] [-long_help] [-bank_index &lt;bank index&gt;] [-bank_name &lt;bank name&gt;] -instance_name &lt;instance name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-bank_index &lt;bank index&gt;</code>	Index of the bank on the mux to be used as the source of the output pins
	<code>-bank_name &lt;bank name&gt;</code>	Name of the bank to be used as the source of the output pins
	<code>-instance_name &lt;instance name&gt;</code>	Name of the Logic Analyzer Interface instance to change
<b>Description</b>	Change the Logic Analyzer Interface output pin's source on the specified instance to use the specified bank.	
<b>Example Usage</b>	<pre># Start a new control sequence. begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path "lai_demo.lai"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Tristate the output pins tristate_output_pin -instance_name "auto_lai_0"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # End the control sequence. end_logic_analyzer_interface_control</pre>	
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>
	TCL_OK	0
		INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: No Logic Analyzer Interface control sequence has been started.
TCL_ERROR	1	ERROR: The specified device is not found.
TCL_ERROR	1	ERROR: The specified hardware is not found.
TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance in the file is not compatible with the instance in the device.
TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The specified bank cannot be found in the Logic Analyzer Interface instance.
TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance cannot be found.
TCL_ERROR	1	ERROR: The version of the specified Logic Analyzer Interface instance is not supported in this release of software.

### [3.1.19.3. end\\_logic\\_analyzer\\_interface\\_control \(:quartus::logic\\_analyzer\\_interface\)](#)

The following table displays information for the `end_logic_analyzer_interface_control` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::logic_analyzer_interface</a> on page 221	
<b>Syntax</b>	<code>end_logic_analyzer_interface_control [-h   -help] [-long_help]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
<b>Description</b>	Terminate the Logic Analyzer Interface control sequence.	
<b>Example Usage</b>	<pre># Start a new control sequence. begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EPIC20 (0x020840DD)" -file_path "lai_demo.lai"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Tristate the output pins tristate_output_pin -instance_name "auto_lai_0"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # End the control sequence. end_logic_analyzer_interface_control</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No Logic Analyzer Interface control sequence has been started.

### 3.1.19.4. get\_current\_state\_of\_output\_pin (::quartus::logic\_analyzer\_interface)

The following table displays information for the `get_current_state_of_output_pin` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::logic_analyzer_interface on page 221		
<b>Syntax</b>	<code>get_current_state_of_output_pin [-h   -help] [-long_help] -instance_name &lt;instance name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-instance_name <instance name>	Name of the Logic Analyzer Interface instance to change	
<b>Description</b>	Query the device to get the current state of the output pins of the specified instance. The result is either the bank name or "tristated".		
<b>Example Usage</b>	<pre># Start a new control sequence. begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path "lai_demo.lai"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Tristate the output pins tristate_output_pin -instance_name "auto_lai_0"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # End the control sequence. end_logic_analyzer_interface_control</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No Logic Analyzer Interface control sequence has been started.
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance in the file is not compatible with the instance in the device.

*continued...*

TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance cannot be found.
TCL_ERROR	1	ERROR: The version of the specified Logic Analyzer Interface instance is not supported in this release of software.

### 3.1.19.5. tristate\_output\_pin (::quartus::logic\_analyzer\_interface)

The following table displays information for the `tristate_output_pin` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::logic_analyzer_interface</a> on page 221		
<b>Syntax</b>	<code>tristate_output_pin [-h   -help] [-long_help] -instance_name &lt;instance name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-instance_name <instance name>	Name of the Logic Analyzer Interface instance to change	
<b>Description</b>	Tristate the output pins of the specified Logic Analyzer Interface instance.		
<b>Example Usage</b>	<pre># Start a new control sequence. begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \[USB-0\]" -device_name "@\1: EPIC20 (0x020840DD)" -file_path "lai_demo.lai"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Change input bank source to the output pins change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # Tristate the output pins tristate_output_pin -instance_name "auto_lai_0"  # Query the output pin state. puts "Current output pin state of instance auto_lai_0:" puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]  # End the control sequence. end_logic_analyzer_interface_control</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No Logic Analyzer Interface control sequence has been started.
	TCL_ERROR	1	ERROR: The specified device is not found.
	TCL_ERROR	1	ERROR: The specified hardware is not found.
	TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance in the file is not compatible with the instance in the device.

*continued...*

TCL_ERROR	1	ERROR: JTAG communication error is detected. It can be caused by the hardware failure or poor signal integrity in the JTAG chain.
TCL_ERROR	1	ERROR: The specified Logic Analyzer Interface instance cannot be found.
TCL_ERROR	1	ERROR: The version of the specified Logic Analyzer Interface instance is not supported in this release of software.

### 3.1.20. ::quartus::misc

The following table displays information for the **::quartus::misc** Tcl package:

<b>Tcl Package and Version</b>	::quartus::misc 1.0
<b>Description</b>	This package contains a set of Tcl functions for performing miscellaneous tasks.
<b>Availability</b>	<p>This package is loaded by default in the following executables:</p> <pre> hdb_debug qacv qpro qpro_sh quartus quartus_cdb quartus_drc quartus_eda quartus_fit quartus_ipc quartus_ipd quartus_ipgenerate quartus_map quartus_sh quartus_si quartus_sim quartus_sta quartus_stp </pre>
<b>Tcl Commands</b>	<pre> checksum (::quartus::misc) on page 227 disable_natural_bus_naming (::quartus::misc) on page 228 enable_natural_bus_naming (::quartus::misc) on page 228 escape_brackets (::quartus::misc) on page 229 foreach_in_collection (::quartus::misc) on page 230 get_collection_size (::quartus::misc) on page 232 get_environment_info (::quartus::misc) on page 232 get_message_count (::quartus::misc) on page 233 init_tk (::quartus::misc) on page 233 load (::quartus::misc) on page 234 load_package (::quartus::misc) on page 234 post_message (::quartus::misc) on page 235 qerror (::quartus::misc) on page 236 qexec (::quartus::misc) on page 236 qexit (::quartus::misc) on page 237 record_tcl_cmd (::quartus::misc) on page 237 stopwatch (::quartus::misc) on page 238 </pre>

#### 3.1.20.1. checksum (::quartus::misc)

The following table displays information for the **checksum** Tcl command:

<b>Tcl Package and Version</b>	Belongs to <b>::quartus::misc</b> on page 227	
<b>Syntax</b>	<code>checksum [-h   -help] [-long_help] [-algorithm &lt;crc32 adler32&gt; ] &lt;input_file&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-algorithm &lt;crc32 adler32&gt;</code>	Option to specify the checksum algorithm. Uses the CRC-32 algorithm by default.

*continued...*

	<input_file>		Option to specify the input file
<b>Description</b>	Returns the checksum value in hexadecimal format based on the specified algorithm which defaults to crc32.		
<b>Example Usage</b>	<pre>set file "one_wire.sof" # Use CRC-32 puts "\$file -&gt; [checksum \$file]" puts "\$file -&gt; [checksum \$file -algorithm crc32]" # Use ADLER-32 puts "\$file -&gt; [checksum \$file -algorithm adler32]"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't read file: <string>. Make sure the file exists and is not a directory, and you have permission to read the file.

### 3.1.20.2. disable\_natural\_bus\_naming (::quartus::misc)

The following table displays information for the disable\_natural\_bus\_naming Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	disable_natural_bus_naming [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help -long_help		
	Short help  Long help with examples and possible return values		
<b>Description</b>	Disables natural bus naming. You may choose to disable natural bus naming to string match patterns such as "in\[024\]". In this example, you are string matching the names "in0", "in2", and "in4". Note that if you disable natural bus naming, then square brackets must be escaped twice (\\\[ or \\\]) so that the strings are interpreted as bus names during a string match, such as: set bus_name "address \\[0\\]" string match [escape_brackets \$bus_name] \$bus_name The "escape_brackets" command escapes "address\\[0\\]" into "address\\\\[0\\\\]". To enable natural bus naming again, type "enable_natural_bus_naming". For more information about natural bus naming, type "enable_natural_bus_naming -h".		
<b>Example Usage</b>	# Disables natural bus naming disable_natural_bus_naming		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.20.3. enable\_natural\_bus\_naming (::quartus::misc)

The following table displays information for the enable\_natural\_bus\_naming Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	enable_natural_bus_naming [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help -long_help		
	Short help  Long help with examples and possible return values		
	<i>continued...</i>		

<b>Description</b>	Enables natural bus naming so that square brackets for bus names do not have to be escaped to prevent Tcl from interpreting them as sub-commands. Bus names have the following format: <bus name>[<bus index>] or <bus name>[*] The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names. Many Quartus Prime Tcl commands allow bus names in their arguments, such as: set_location_assignment -to address[10] Pin_M20 If natural bus naming is enabled, you can just use address[10] instead of having to escape the square brackets into address[10]. There are also Quartus Prime Tcl commands that take Tcl string match patterns in their arguments, such as: get_all_instance_assignments -name location -to address[10] Since Tcl string matching takes string patterns containing special characters from the set "*?\[" as values, address[10] would be interpreted incorrectly. By enabling natural bus naming, these Tcl commands will automatically detect address[10] as a bus name so that you don't have to doubly escape the brackets into address\\\[10\\]\]. To disable natural bus naming, type "disable_natural_bus_naming". For more information on the effects of disabling natural bus naming, type "disable_natural_bus_naming -h".						
<b>Example Usage</b>	# Enables natural bus naming enable_natural_bus_naming						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.20.4. escape\_brackets (::quartus::misc)

The following table displays information for the escape\_brackets Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	escape_brackets [-h   -help] [-long_help] <str>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<str>	String to escape	
<b>Description</b>	Escapes square brackets in bus name patterns for use in string matching. Also escapes the escape character; for example, the string "\\" is escaped into "\\\". Note that natural bus naming is supported by default. This means that bus names, not bus name patterns, are automatically detected and do not need to be escaped by using this command. Bus names have the following format: <bus name>[<bus index>] or <bus name>[*] The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names. All other uses of square brackets must be escaped if you do not intend to use the brackets as part of a string pattern in a string match. For example, the bus name pattern "a\[0-2\]" must be escaped using this "escape_brackets" command since the "0-2" does not satisfy the <bus index> requirement to be a bus name. Many Quartus Prime Tcl commands allow string matching in option arguments. A common error is using bus name patterns in these arguments, such as: address\[0-2\] Square brackets for bus name patterns must already be preceded by an escape character (\[ or \]) to prevent Tcl from interpreting them as sub-commands. String matching, however, also uses square brackets to match any character between the brackets. The previous example, when used as a string match pattern, searches for the string patterns address0, address1, and address2. It does not search for address[0], address[1], and address[2]. Therefore, for arguments that support string matching, square brackets must be escaped twice (\\\\[ or \\\]\]) so that the strings are interpreted as bus name patterns. For example, to search for address[0], address[0], and address[2], type the following string match pattern: address\\\[0-2\\]\\] or, equivalently, "address[escape_brackets \\\[0-2\\]\\[escape_brackets \\]]" Quartus Prime Tcl commands do not convert bus name patterns automatically, since they cannot determine if the string is intended as a bus name pattern or a regular string match pattern. Therefore, "escape_brackets" is provided for your convenience. You may choose to disable natural bus naming in order to string match patterns such as "in\[024\]". In this example, you are string matching the names "in0", "in2", and "in4". To disable natural bus naming, type "disable_natural_bus_naming". Note that if you disable natural bus naming, then square brackets must be escaped twice (\\\\[ or \\\]\]) so that the strings are interpreted as bus names during a string match, such as: set bus_name "address\[0\]" string match [escape_brackets		

*continued...*

	\$bus_name] \$bus_name The "escape_brackets" command escapes "address\[0\]" into "address\\\[0\\\[0\]". To enable natural bus naming again, type "enable_natural_bus_naming". For more information about natural bus naming, type "enable_natural_bus_naming -h".						
<b>Example Usage</b>	<pre># Get all location assignments for bus address[] set address_names address[*] set address_locations [get_all_instance_assignments -to \$address_names] -name LOCATION  # Get location assignment for bus address[0] set address_names address[0] set address_locations [get_all_instance_assignments -to \$address_names] -name LOCATION  # Get location assignments for bus address[0], # address[1], and address[2] set address_names address[0-2] set address_locations [get_all_instance_assignments -to [escape_brackets \$address_names]] - name LOCATION</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.20.5. foreach\_in\_collection (::quartus::misc)

The following table displays information for the `foreach_in_collection` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227	
<b>Syntax</b>	<code>foreach_in_collection [-h   -help] [-long_help] &lt;variable_name&gt; &lt;collection&gt; &lt;body&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>&lt;variable_name&gt;</code>	Variable name
	<code>&lt;collection&gt;</code>	Collection
	<code>&lt;body&gt;</code>	Body
<b>Description</b>	Accesses each element of a collection. Some Tcl commands return a collection. The following table shows examples of commands that return a collection: Tcl package Tcl commands (returning a collection) ----- ::quartus::project get_all_quartus_defaults get_all_global_assignments get_all_instance_assignments get_all_parameters get_names assignment_group (only for the "-get_members" and "-get_exceptions" options) ::quartus::chip_editor get_nodes get_iports get_oports The command is used in the following format: <code>foreach_in_collection &lt;variable name&gt; &lt;collection&gt; { # This is the body of "foreach_in_collection" ... }</code> Unlike a Tcl list, a collection is a container specific to the Quartus II software, whose elements can be accessed by using the "foreach_in_collection" command.	
<b>Example Usage</b>	<pre>## Get a collection of global assignments set collection_of_global_assignments [get_all_global_assignments -name *] ## Display the collection string representation puts \$collection_of_global_assignments ## Iterate through the collection and display ## the information for each global assignment foreach_in_collection global \$collection_of_global_assignments {     set sect_id [lindex \$global 0]     set name [lindex \$global 1]     set value [lindex \$global 2]      ## Now, display the content of the global assignment     puts "Section ID (\$sect_id)"     puts "Assignment Name (\$name)"     puts "Assignment Value (\$value)" }  ## Get a collection of instance assignments set collection_of_instance_assignments [get_all_instance_assignments -name *] ## Display the collection string representation puts \$collection_of_instance_assignments</pre>	

*continued...*

```

## Iterate through the collection and display
## the information for each instance assignment
foreach_in_collection instance $collection_of_instance_assignments {

    set sect_id [lindex $instance 0]
    set src [lindex $instance 1]
    set dest [lindex $instance 2]
    set name [lindex $instance 3]
    set value [lindex $instance 4]

    ## Now, display the content of the instance assignment
    puts "Section ID ($sect_id)"
    puts "Source ($src)"
    puts "Destination ($dest)"
    puts "Assignment Name ($name)"
    puts "Assignment Value ($value)"
}

## Get a collection of parameters
set collection_of_parameters [get_all_parameters -name *]
## Display the collection string representation
puts $collection_of_parameters
## Iterate through the collection and display
## the information for each parameter
foreach_in_collection parameter $collection_of_parameters {

    set dest [lindex $parameter 0]
    set name [lindex $parameter 1]
    set value [lindex $parameter 2]

    ## Now, display the content of the parameter
    puts "Destination ($dest)"
    puts "Parameter Name ($name)"
    puts "Parameter Value ($value)"
}

## Get a collection of all node name ids from a successful
## compilation
set collection_of_name_ids [get_names -filter *]
## Display the collection string representation
puts $collection_of_name_ids
## Iterate through the collection and display
## the information for each name id
foreach_in_collection name_id $collection_of_name_ids {

    set parent_name_id [get_name_info -info parent_name_id $name_id]
    set base_name [get_name_info -info base_name $name_id]
    set entity_name [get_name_info -info entity_name $name_id]
    set instance_name [get_name_info -info instance_name $name_id]
    set full_path [get_name_info -info full_path $name_id]
    set short_full_path [get_name_info -info short_full_path $name_id]
    set node_type [get_name_info -info node_type $name_id]
    set creator [get_name_info -info creator $name_id]
    set signaltapii [get_name_info -info signaltapii $name_id]
    set file_location [get_name_info -info file_location $name_id]

    ## Now, display information about the name
    puts "Parent Name Id ($parent_name_id)"
    puts "Base Name ($base_name)"
    puts "Entity Name ($entity_name)"
    puts "Instance Name ($instance_name)"
    puts "Full Path ($full_path)"
    puts "Short Full Path ($short_full_path)"
    puts "Node Type ($node_type)"
    puts "Creator ($creator)"
    puts "Signaltapii ($signaltapii)"
    puts "File location ($file_location)"
}

# Display the members of a particular assignment group named "tgt"
foreach_in_collection member [assignment_group "tgt" -get_members] {

    # Print the name of the member
    puts $member
}

# Display the exception to a particular assignment group named "tgt"
foreach_in_collection exception [assignment_group "tgt" -get_exceptions] {

    # Print the name of the exception
    puts $exception
}

```

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.20.6. get\_collection\_size (::quartus::misc)

The following table displays information for the `get_collection_size` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>get_collection_size [-h   -help] [-long_help] [ &lt;collection&gt; ]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>&lt;collection&gt;</code>	Collection	
<b>Description</b>	Returns the size of a collection. Unlike a Tcl list, a collection is a container specific to the Quartus Prime software, whose elements can be accessed by using the "foreach_in_collection" command.		
<b>Example Usage</b>	<pre>## Displays the number of global assignments project_open chiptrip set num_global_asgns [get_collection_size [get_all_global_assignments -name {*}]] puts \$num_global_asgns project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Collection does not exist with name: <string>. Specify the collection name returned by a Tcl command that supports foreach_in_collection. Note a valid collection name can become invalid if the variable holding the collection goes out of scope, as well as a result of some built-in TCL commands, for example 'string length'.
	TCL_ERROR	1	ERROR: Nested calls to foreach_in_collection with the same collection name <string> are not allowed. Specify a different collection name for the other collection.

### 3.1.20.7. get\_environment\_info (::quartus::misc)

The following table displays information for the `get_environment_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>get_environment_info [-h   -help] [-long_help] [-num_logical_processors] [-num_physical_processors] [-operating_system]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-num_logical_processors</code>	Option to return the number of available logical processors (cores including hyper-threading)	
	<code>-num_physical_processors</code>	Option to return the number of available physical processors (sockets)	
	<code>-operating_system</code>	Option to return the operating system name	
<b>Description</b>	Returns information about the system environment depending on the options specified.		
<b>Example Usage</b>	<pre># Get the number of physical processors available on my computer. get_environment_info -num_physical_processors</pre>		
			<i>continued...</i>

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: At least one option is required. Specify at least one option.
	TCL_ERROR	1	ERROR: Multiple options used. Choose only one option for this command.

### 3.1.20.8. get\_message\_count (::quartus::misc)

The following table displays information for the `get_message_count` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	<code>get_message_count [-h   -help] [-long_help] -type &lt;info extra_info warning critical_warning error&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-type <info extra_info warning critical_warning error>	Type of message	
<b>Description</b>	Get a count of messages of a specific message type. The message type can be "info", "warning", "critical_warning", or "error".		
<b>Example Usage</b>	<pre># Get count of error messages get_message_count -type error  # Get count of warning messages get_message_count -type warning  # Get count of critical warning messages get_message_count -type critical_warning</pre>		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal message type: <string>. Specify info, warning, critical_warning, or error.
	TCL_ERROR	1	ERROR: Missing required positional argument: <string>. Specify the <string> argument.
	TCL_ERROR	1	ERROR: You specified <string> arguments to the -args option. However, you can pass a maximum of <string> arguments.

### 3.1.20.9. init\_tk (::quartus::misc)

The following table displays information for the `init_tk` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	<code>init_tk [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Initializes a Tk window. If you are using Tk functionality in Tcl, you must run this command first before running any Tcl scripts.		

*continued...*

<b>Example Usage</b>	<pre># Initialize the Tk library init_tk  # Create a top level and add a title toplevel .top wm title .top "Hello World"  # Add widgets button .top.hello -text Hello -command {puts stdout "Hello, World!"} pack .top.hello -padx 20 -pady 10  # Without "tkwait", the script finishes at this point and the # window is destroyed. The "tkwait" command prevents the # script from finishing until the you close the window. tkwait window .top</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.20.10. load (::quartus::misc)

The following table displays information for the `load` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>load [-h   -help] [-long_help] &lt;load_args&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<load_args>	Arguments to load	
<b>Description</b>	Loads machine code and initializes new commands. This command works exactly like Tcl's native "load" command.		
<b>Example Usage</b>	Refer to documentation for Tcl's native "load" command at the Tcl/Tk web site at <a href="http://www.tcl.tk">www.tcl.tk</a> .		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't load library: <string>. The operating system reports the following error: <string>

### 3.1.20.11. load\_package (::quartus::misc)

The following table displays information for the `load_package` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>load_package [-h   -help] [-long_help] [-version &lt;version number&gt;] &lt;package name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-version <version number>	Option to specify the Quartus Prime Tcl package version to load	
	<package name>	Name of Quartus Prime Tcl package to load	

*continued...*

<b>Description</b>	Loads the specified Quartus Prime Tcl package with the specified version number. If you do not specify the "-version" option, the latest version is loaded by default. The Quartus Prime Tcl package names have the "::quartus::" prefix, such as "::quartus::project". For convenience, you can omit the "::quartus::" prefix when you use the <package name> argument. This command is similar to the "package require" command. The advantage of using "load_package" is that you can alternate freely between different versions of the same package. For example, if you loaded version 2.0, and now want to load version 1.0, you can type: "load_package -version 1.0 <package name>".		
<b>Example Usage</b>	<pre># Load version 1.0 of the ::quartus::project package load_package project -version 1.0  # Load version 2.0 of the ::quartus::project package load_package project -version 2.0</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Tcl package <string> does not exist. Specify an available Quartus Prime Tcl package. Type help for a list of available Quartus Prime Tcl packages.
	TCL_ERROR	1	ERROR: Tcl package <string> version <string> does not exist. Specify an available Quartus Prime Tcl package. Type help for a list of available Quartus Prime Tcl packages.
	TCL_ERROR	1	ERROR: Tcl package <string> is only available in Quartus Pro Edition
	TCL_ERROR	1	ERROR: Tcl package <string> is only available in Quartus Standard Edition

### 3.1.20.12. post\_message (::quartus::misc)

The following table displays information for the post\_message Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227	
<b>Syntax</b>	<pre>post_message [-h   -help] [-long_help] [-type &lt;info extra_info warning  critical_warning error&gt;] [ &lt;string&gt; ]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-type <info extra_info warning  critical_warning error>	Type of message to display
	<string>	Message to be displayed
<b>Description</b>	Displays messages and sub-messages of the specified type. The message type can be "info", "warning", "critical_warning", or "error". If you do not use the -type option, the default message type is "info". The "extra_info" type is deprecated. Messages with this type will not be displayed. Use the "info" type instead. Use the -submsgs option to group messages indented under a message. The -submsgs option posts each string in a Tcl list of strings as a sub-message. The sub-messages have the same message type as the main message.	
<b>Example Usage</b>	<pre># Display an error message post_message -type error "Can't open file test.tcl"  # Display an info message post_message "Generated output file: test.out" # OR post_message -type info "Generated output file: test.out"  # Display an info message with a sub-message post_message "Generated output file: test.out" -submsgs [list "Output file saved in project directory"]  # Display a warning message</pre>	

*continued...*

	<pre>post_message -type warning "Defaulting fmax to 155.55mhz" # Display a critical warning message post_message -type critical_warning "Invalid fmax was specified - defaulting to 155.55mhz"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Error(s) found while processing handle: <string>. Make sure you specified a valid Quartus Prime message handle for the -<string> option and passed the correct number of arguments to the -<string> option. Also check that you passed the correct message type to the -<string> option.
	TCL_ERROR	1	ERROR: Illegal message type: <string>. Specify info, warning, critical_warning, or error.
	TCL_ERROR	1	ERROR: Missing required positional argument: <string>. Specify the <string> argument.
	TCL_ERROR	1	ERROR: You specified <string> arguments to the -args option. However, you can pass a maximum of <string> arguments.

### 3.1.20.13. qerror (::quartus::misc)

The following table displays information for the `qerror` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>qerror [-h   -help] [-long_help] [-error_null] [-over_malloc] [-qt_segfault] [-std_segfault]</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-error_null		Option to error by dereferencing null
	-over_malloc		Option to error by malloc'ing INT_MAX
	-qt_segfault		Accesses an element at an invalid index in a QT container
	-std_segfault		Accesses an element at an invalid index in a std container
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.20.14. qexec (::quartus::misc)

The following table displays information for the `qexec` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>qexec [-h   -help] [-long_help] &lt;command&gt;</code>		
<b>Arguments</b>	-h   -help		Short help
	<b>continued...</b>		

	-long_help	Long help with examples and possible return values			
	<command>	Command			
<b>Description</b>	Runs the specified shell command from within a Tcl shell or script. Usage for this command is as follows: qexec "<command>"				
<b>Example Usage</b>	qexec ls				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.20.15. qexit (::quartus::misc)

The following table displays information for the qexit Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227				
<b>Syntax</b>	qexit [-h   -help] [-long_help] [-error] [-success]				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-error	Option to exit with an equivalent Quartus Prime error code			
	-success	Option to exit with an equivalent Quartus Prime success code			
<b>Description</b>	Exits the Quartus Prime software. The Quartus Prime Tcl command "qexit" is equivalent to the Tcl command "exit". When used with a particular option, this command exits the Quartus Prime software with the corresponding Quartus Prime exit code. For example, typing "qexit -success" is equivalent to typing "exit 0". When the "-success" option is specified, the corresponding Quartus Prime exit code is "0". If no option is specified, the default exit code is the same as for the "-success" option.				
<b>Example Usage</b>	1) To exit the Quartus Prime software with an equivalent Quartus Prime success code, type: <pre>qexit -success</pre> 2) To exit the Quartus Prime software with an equivalent Quartus Prime error code, type: <pre>qexit -error</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.20.16. record\_tcl\_cmd (::quartus::misc)

The following table displays information for the record\_tcl\_cmd Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::misc on page 227		
<b>Syntax</b>	record_tcl_cmd [-h   -help] [-long_help] [-filename <file name>] <start_end>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-filename <file name>	Option to specify an user filename	
	<start_end>	To start or end recording	
<i>continued...</i>			

<b>Description</b>	Start or end to record Tcl commands.		
<b>Example Usage</b>	1) Type "record_tcl_cmd" start -filename user.rec To start record Tcl commands with filename "user.rec"		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.20.17. stopwatch (::quartus::misc)

The following table displays information for the `stopwatch` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::misc</a> on page 227		
<b>Syntax</b>	<code>stopwatch [-h   -help] [-long_help] [-lap_time] [-number_format] [-reset] [-start]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-lap_time	Option to get the lap time	
	-number_format	Option to show the lap time in seconds without appending the "s", i.e. seconds, character	
	-reset	Option to reset the stopwatch	
	-start	Option to start the stopwatch	
<b>Description</b>	Provides a stopwatch interface.		
<b>Example Usage</b>	<pre># Begin the stopwatch stopwatch -start exec sleep 1 # Get the lap_time puts [stopwatch -lap_time] exec sleep 1 # Get the lap time puts [stopwatch -lap_time] # Reset the stopwatch stopwatch -reset</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.21. ::quartus::names

The following table displays information for the `::quartus::names` Tcl package:

<b>Tcl Package and Version</b>	::quartus::names 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is available for loading in the following executables: <pre>qpro_sh quartus_asm quartus_cdb quartus_edt quartus_fit quartus_map quartus_pow quartus_sh quartus_sta quartus_syn</pre>

*continued...*

<b>Tcl Commands</b>	<a href="#">get_assignment (::quartus::names) on page 239</a> <a href="#">set_assignment (::quartus::names) on page 239</a>
---------------------	--

### 3.1.21.1. get\_assignment (::quartus::names)

The following table displays information for the `get_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::names</a> on page 238		
<b>Syntax</b>	<code>get_assignment [-h   -help] [-long_help] -dict -name &lt;ASSIGNMENT_NAME&gt; [-to &lt;ASSIGNMENT_TO&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-dict	Indicates this command should return its result as a Tcl dict object.	
	-name <ASSIGNMENT_NAME>	The assignment name to get the value of.	
	-to <ASSIGNMENT_TO>	The node to retrieve the assignment value from (omitting this will retrieve the global assignment value, if any).	
<b>Description</b>	Gets the assignment's value. If -to is specified, tries to get the assignment value on the given node, as long as the assignment targeted the node using its -to field. If -to is not specified, the global value is returned (if any). If the assignment is not found, a Tcl error is produced. If the assignment is found and the -dict option is given, a Tcl dict object is returned with the assignment 'name' (the given input), and the 'value' retrieved from the found assignment. In addition, if the assignment has a 'to' or 'from' field, those fields will also exist in the returned dict.		
<b>Example Usage</b>	# Returns fitter seed <code>get_assignment SEED</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No assignment matching the name <string> was found.

### 3.1.21.2. set\_assignment (::quartus::names)

The following table displays information for the `set_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::names</a> on page 238		
<b>Syntax</b>	<code>set_assignment [-h   -help] [-long_help] -name &lt;ASSIGNMENT_NAME&gt; [-to &lt;INSTANCE_NAME&gt; ] -value &lt;VALUE&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <ASSIGNMENT_NAME>	The assignment name to set.	
	-to <INSTANCE_NAME>	The node to which the assignment should be applied.	
	-value <VALUE>	The value to set the assignment to.	
<b>Description</b>	Sets the assignment to the given value. If -to is specified, tries to set the assignment value to the given node. If -to is not specified, this applies to global assignments.		

*continued...*

<b>Example Usage</b>	set_assignment SEED 9		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No assignment matching the name <string> was found.

### 3.1.22. ::quartus::periph

The following table displays information for the **::quartus::periph** Tcl package:

<b>Tcl Package and Version</b>	::quartus::periph 1.0
<b>Description</b>	This package contains the set of Tcl functions for interacting with the Interface Planner plans.
<b>Availability</b>	This package is available for loading in the following executable: quartus_fit
<b>Tcl Commands</b>	<pre>blueprint::initialize (::quartus::periph) on page 245 blueprint::shutdown (::quartus::periph) on page 253 periph::check_plan (::quartus::periph) on page 240 periph::get_cell_info (::quartus::periph) on page 241 periph::get_cells (::quartus::periph) on page 242 periph::get_location_info (::quartus::periph) on page 243 periph::get_placement_info (::quartus::periph) on page 244 periph::get_placements (::quartus::periph) on page 244 periph::load_floorplan (::quartus::periph) on page 245 periph::place_cells (::quartus::periph) on page 246 periph::remove_invalid_reports (::quartus::periph) on page 247 periph::report_all (::quartus::periph) on page 247 periph::report_cell_connectivity (::quartus::periph) on page 248 periph::report_cell_placement_reasons (::quartus::periph) on page 248 periph::report_cells (::quartus::periph) on page 249 periph::report_clocks (::quartus::periph) on page 249 periph::report_legal_cell_locations (::quartus::periph) on page 250 periph::report_location_types (::quartus::periph) on page 250 periph::report_locations (::quartus::periph) on page 250 periph::report_regions (::quartus::periph) on page 251 periph::report_summary (::quartus::periph) on page 251 periph::reset_plan (::quartus::periph) on page 252 periph::save_floorplan (::quartus::periph) on page 252 periph::set_clock_type (::quartus::periph) on page 253 periph::undo_last_placement (::quartus::periph) on page 254 periph::unplace_cells (::quartus::periph) on page 254 periph::update_pdw (::quartus::periph) on page 255 periph::update_plan (::quartus::periph) on page 255 periph::write_plan (::quartus::periph) on page 256</pre>

#### 3.1.22.1. periph::check\_plan (::quartus::periph)

The following table displays information for the **periph::check\_plan** Tcl command:

<b>Tcl Package and Version</b>	Belongs to <b>::quartus::periph</b> on page 240	
<b>Syntax</b>	periph::check_plan [-h   -help] [-long_help]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
<b>Description</b>	Checks the legality of the current plan	
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan</pre>	

*continued...*

	<pre>periph::check_plan project_close</pre>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.22.2. periph::get\_cell\_info (::quartus::periph)

The following table displays information for the `periph::get_cell_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	<code>periph::get_cell_info [-h   -help] [-long_help] [-children] [-guide_cell_id] [-ip_type] [-links] [-location] [-name] [-parent] [-type] &lt;cell_id&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-children	Return the the cell id of the children cells	
	-guide_cell_id	Returns the guide cell's elem_id	
	-ip_type	Returns the IP type if the cell is an IP cell or an empty string otherwise	
	-links	Return the given design element's connections to other cells	
	-location	Returns the location ID if the cell is placed or an empty string otherwise	
	-name	Return the cell name of the cell id	
	-parent	Return the the cell id of the parent cells	
	-type	Return the the type of the cell	
<b>Description</b>	<cell_id>	Single cell id	
	Gets information about the specified cell (referenced by cell ID). You can obtain cell using the <code>periph::get_cells</code> Tcl command.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan foreach cell [periph::get_cells -type IO_CLUSTER] {     puts "Found cell ID \$cell named [periph::get_cell_info -name \$cell]" } blueprint::shutdown project_close</pre>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied cell id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one cell ID must be supplied, but no cell IDs were supplied
	TCL_ERROR	1	ERROR: <string> cell IDs were expected but <string> were supplied

### 3.1.22.3. periph::get\_cells (::quartus::periph)

The following table displays information for the `periph::get_cells` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::get_cells [-h   -help] [-long_help] [-atom_only] [-instance_only] [-ip_only] [-num_location &lt;num_location&gt;] [-physical_only] [-placed] [-toplevel_only] [-type &lt;type&gt;] [-unplaced] [ &lt;filter&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-atom_only	Return only atom cells	
	-instance_only	Return only instance cells	
	-ip_only	Return only instance cells that are IP instances	
	-num_location <num_location>	Specify a specific number of available locations the cell should have	
	-physical_only	Return only physical cells	
	-placed	Return only placed cells	
	-toplevel_only	Return only toplevel cells	
	-type <type>	Return only cells of the given types	
	-unplaced	Return only unplaced cells	
<b>Description</b>	<filter>		
	Returns a list of cells IDs in the design. All cell names in the collection match the specified pattern. Wildcards can be used to select multiple cells at once. When you use the wildcard matching, use pipe characters to separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. When this matching scheme is enabled, the specified pattern is matched against absolute cell names: the names that include the entire hierarchical path. A full cell name can contain multiple pipe characters in it to reflect the hierarchy. All hierarchy levels in the pattern are matched level by level. Any included wildcards refer to only one hierarchical level. For example, "*" and "* *" produce different collections since they refer to the highest hierarchical level and second highest hierarchical level respectively.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan  foreach cell [periph::get_cells -type IO_CLUSTER] {     puts "Found cell ID \$cell named [periph::get_cell_info -name \$cell]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied number of locations <string> is invalid.
	TCL_ERROR	1	ERROR: <string> number of locations were expected but <string> were supplied

### 3.1.22.4. periph::get\_location\_info (::quartus::periph)

The following table displays information for the `periph::get_location_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::periph</code> on page 240		
<b>Syntax</b>	<code>periph::get_location_info [-h   -help] [-long_help] [-children] [-gid] [-name] [-parents] [-placed_cells] [-properties] [-type] &lt;location_id&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-children</code>	Query the children location IDs	
	<code>-gid</code>	Query the gid of the location IDs	
	<code>-name</code>	Return the location name of the location id	
	<code>-parents</code>	Query the parent location IDs	
	<code>-placed_cells</code>	Return the placed cells at the location id	
	<code>-properties</code>	Return the device location properties in json	
	<code>-type</code>	Return the location type of the location id	
<b>Description</b>	<location_id>		
	location id		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan  foreach cell [periph::get_cells -placed] {     puts "Found cell ID \$cell named [periph::get_cell_info -name \$cell] placed in location [periph::get_cell_info -location \$cell] named [periph::get_location_info -name [periph::get_cell_info -location \$cell]]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied location id <string> is invalid.
	TCL_ERROR	1	ERROR: The supplied type <string> is invalid.
	TCL_ERROR	1	ERROR: At least one device location ID must be supplied, but no location IDs were supplied
	TCL_ERROR	1	ERROR: At least one type must be supplied, but no types were supplied
	TCL_ERROR	1	ERROR: <string> location IDs were expected but <string> were supplied
	TCL_ERROR	1	ERROR: <string> types were expected but <string> were supplied

### 3.1.22.5. periph::get\_placement\_info (::quartus::periph)

The following table displays information for the periph::get\_placement\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::get_placement_info [-h   -help] [-long_help] [-placement] <placement_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-placement	Return the placement as a list of cell/id pairs	
	<placement_id>	Single placement id	
<b>Description</b>	Return information about a given placement		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied placement id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one placement ID must be supplied, but no placement IDs were supplied
	TCL_ERROR	1	ERROR: <string> placement IDs were expected but <string> were supplied

### 3.1.22.6. periph::get\_placements (::quartus::periph)

The following table displays information for the periph::get\_placements Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::get_placements [-h   -help] [-long_help] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<cell_id>	Single cell id	
<b>Description</b>	Returns a vector of placements for the supplied cell		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied cell id <string> is not a placeable cell.
	TCL_ERROR	1	ERROR: The supplied cell id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one cell ID must be supplied, but no cell IDs were supplied
	TCL_ERROR	1	ERROR: <string> cell IDs were expected but <string> were supplied

### 3.1.22.7. blueprint::initialize (::quartus::periph)

The following table displays information for the `blueprint::initialize` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>blueprint::initialize [-h   -help] [-long_help] [-load_compiler_snapshot] [-read_settings_files &lt;on off&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-load_compiler_snapshot</code>	Initialize the Interface Planner placement state with the latest compiler snapshot. Use <code>periph::compiler_snapshot_exists</code> to check if a compiler snapshot can be loaded	
	<code>-read_settings_files &lt;on off&gt;</code>	Option to identify that settings files should be read from disk	
<b>Description</b>	Initialize the Interface Planner planning engine. Initialization consists of loading all required databases from disk for the design and device. Once initialization is complete, the constraints have not yet been applied and must be done before placement can begin. After the assignments have been created, they can be applied to the project by running the <code>plan::update_platform</code> Tcl command.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.8. periph::load\_floorplan (::quartus::periph)

The following table displays information for the `periph::load_floorplan` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::load_floorplan [-h   -help] [-long_help] -filename &lt;filename&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-filename &lt;filename&gt;</code>	Filename to load	
<b>Description</b>	Load the floorplan from a Interface Planner floorplan file		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan periph::place_cells -unplaced_cells periph::save_floorplan -filename onewire_blueprint_floorplan.plan</pre>		

*continued...*

	<pre>periph::load_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.9. periph::place\_cells (::quartus::periph)

The following table displays information for the periph::place\_cells Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	<pre>periph::place_cells [-h   -help] [-long_help] [-cell_location &lt;cell_location&gt;] [-cells &lt;cells&gt;] [-dont_revert_on_fail] [-fixed_cells] [-placement &lt;placement&gt;] [-unplaced_cells]</pre>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cell_location <cell_location>	Cell location id pair to place cells into	
	-cells <cells>	One or more cell ids	
	-dont_revert_on_fail	Option to specify that the best partial placement should be saved on the undo stack upon a placement failure	
	-fixed_cells	Place all unplaced cells	
	-placement <placement>	Place cells according to a placement. A placement is a special object that comes from the periph::get_placements Tcl command	
	-unplaced_cells	Place all unplaced cells	
<b>Description</b>	Performs a placement on the supplied cells		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan periph::place_cells -unplaced_cells periph::check_plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied cell id <string> is invalid.
	TCL_ERROR	1	ERROR: The supplied cell id / location id pair <string> is invalid.
	TCL_ERROR	1	ERROR: The supplied location id <string> is invalid.
	TCL_ERROR	1	ERROR: The supplied placement id <string> is invalid.
	TCL_ERROR	1	ERROR: At least one cell ID must be supplied, but no cell IDs were supplied

*continued...*

TCL_ERROR	1	ERROR: At least one device location ID must be supplied, but no location IDs were supplied
TCL_ERROR	1	ERROR: At least one placement ID must be supplied, but no placement IDs were supplied
TCL_ERROR	1	ERROR: <string> cell IDs were expected but <string> were supplied
TCL_ERROR	1	ERROR: <string> location IDs were expected but <string> were supplied
TCL_ERROR	1	ERROR: <string> placement IDs were expected but <string> were supplied

### 3.1.22.10. periph::remove\_invalid\_reports (::quartus::periph)

The following table displays information for the periph::remove\_invalid\_reports Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::remove_invalid_reports [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Remove all invalid report		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.11. periph::report\_all (::quartus::periph)

The following table displays information for the periph::report\_all Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_all [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Create all default summary reports		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.12. periph::report\_cell\_connectivity (::quartus::periph)

The following table displays information for the periph::report\_cell\_connectivity Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_cell_connectivity [-h   -help] [-long_help] [-fanins] [-fanouts] [-panel_name <name>] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fanins	Report only the fanins of the cell	
	-fanouts	Report only the fanouts of the cell	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	<cell_id>	Single cell id	
<b>Description</b>	Creates a report of the connectivity for a cell.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.13. periph::report\_cell\_placement\_reasons (::quartus::periph)

The following table displays information for the periph::report\_cell\_placement\_reasons Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_cell_placement_reasons [-h   -help] [-long_help] [-panel_name <name>] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	<cell_id>	Single cell id	
<b>Description</b>	Creates a report of all the locations a particular cell can be placed and the reasons it cannot be placed there		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.14. periph::report\_cells (::quartus::periph)

The following table displays information for the `periph::report_cells` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	<code>periph::report_cells [-h   -help] [-long_help] [-name &lt;name&gt;] [-panel_name &lt;name&gt;] [-placed] [-type &lt;type&gt;] [-unplaced]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	Filter the list of placed cells specifying a name. Wildcards are supported.	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-placed	Report the list of placed cells	
	-type <type>	Filter the list of placed cells specifying a list of types	
	-unplaced	Report the list of unplaced cells	
<b>Description</b>	Returns a list of periphery cells based on the specified criteria.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.15. periph::report\_clocks (::quartus::periph)

The following table displays information for the `periph::report_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	<code>periph::report_clocks [-h   -help] [-long_help] [-panel_name &lt;name&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	Show the signals that are using low-skew routing networks (clock networks) in the device. If applicable, also show any signals that were considered for automatic clock network promotion, but were not promoted.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.16. periph::report\_legal\_cell\_locations (::quartus::periph)

The following table displays information for the periph::report\_legal\_cell\_locations Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_legal_cell_locations [-h   -help] [-long_help] [-panel_name <name>] <cell_id>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	<cell_id>	Single cell id	
<b>Description</b>	Creates a report of the legal periphery cell locations of a cell		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.17. periph::report\_location\_types (::quartus::periph)

The following table displays information for the periph::report\_location\_types Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_location_types [-h   -help] [-long_help] [-panel_name <name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	Creates a report of the location types in the periphery		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.18. periph::report\_locations (::quartus::periph)

The following table displays information for the periph::report\_locations Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::report_locations [-h   -help] [-long_help] [-panel_name <name>] <type>		
<b>Arguments</b>	-h   -help	Short help	
	<i>continued...</i>		

	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	<type>	location type to query	
<b>Description</b>	Creates a report of the locations for the requested type in the periphery		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
		TCL_OK	0

### 3.1.22.19. periph::report\_regions (::quartus::periph)

The following table displays information for the `periph::report_regions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::report_regions [-h   -help] [-long_help] [-panel_name &lt;name&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
		TCL_OK	0

### 3.1.22.20. periph::report\_summary (::quartus::periph)

The following table displays information for the `periph::report_summary` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::report_summary [-h   -help] [-long_help] [-panel_name &lt;name&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
		TCL_OK	0

### 3.1.22.21. periph::reset\_plan (::quartus::periph)

The following table displays information for the periph::reset\_plan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::reset_plan [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Reverts the current design to be unplaced and without assignments applied		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan periph::reset_plan blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.22. periph::save\_floorplan (::quartus::periph)

The following table displays information for the periph::save\_floorplan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::save_floorplan [-h   -help] [-long_help] -filename <filename>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-filename <filename>	Filename to write to	
<b>Description</b>	Write the Interface Planner floorplan that can be reloaded in Interface Planner		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan set io_cells [periph::get_cells -unplaced -type IO_CLUSTER] periph::place_cells -cells \$io_cells periph::save_floorplan -filename onewire_blueprint_floorplan.plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.23. periph::set\_clock\_type (::quartus::periph)

The following table displays information for the `periph::set_clock_type` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::set_clock_type [-h   -help] [-long_help] [-cell_id &lt;cell_id&gt;] [-cell_name &lt;cell_name&gt;] -type &lt;type&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-cell_id &lt;cell_id&gt;</code>	Single cell id	
	<code>-cell_name &lt;cell_name&gt;</code>	Cell name, returned from <code>periph::get_cell_info -name</code>	
	<code>-type &lt;type&gt;</code>	The type of the clock type	
<b>Description</b>	This command currently contains no help description.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The supplied clock cell id <i>&lt;string&gt;</i> is invalid.
	TCL_ERROR	1	ERROR: The supplied clock type <i>&lt;string&gt;</i> is invalid.
	TCL_ERROR	1	ERROR: At least one clock type must be supplied, but no clock types were supplied
	TCL_ERROR	1	ERROR: <i>&lt;string&gt;</i> clock types were expected but <i>&lt;string&gt;</i> were supplied

### 3.1.22.24. blueprint::shutdown (::quartus::periph)

The following table displays information for the `blueprint::shutdown` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>blueprint::shutdown [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Shutdown Interface Planner.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan blueprint::shutdown  project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.25. periph::undo\_last\_placement (::quartus::periph)

The following table displays information for the periph::undo\_last\_placement Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::undo_last_placement [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Undo the last placement or unplacement operation		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.26. periph::unplace\_cells (::quartus::periph)

The following table displays information for the periph::unplace\_cells Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::unplace_cells [-h   -help] [-long_help] [-cells <cells>] [-placed_cells]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cells <cells>	One or more cell ids	
	-placed_cells	Unplace all placed cells	
<b>Description</b>	Removes the placement from the specified cells. Any constraints for the cells remain, but the cell no longer has a placement.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan periph::unplace_cells -placed_cells periph::check_plan project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.27. periph::update\_pdpw (::quartus::periph)

The following table displays information for the `periph::update_pdpw` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::update_pdpw [-h   -help] [-long_help] [-assignments] [-placement]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-assignments</code>	Indicates assignment model needs updating	
	<code>-placement</code>	Indicates placement needs updating	
<b>Description</b>	Update everything that needs updating in pdpw. This essentially sends a single TCL command to pdpw to update everything as needed. Used in the TCL proc source wrapper only.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_pdpw -pdp_state [blueprint_internal::get_pdp_state]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.28. periph::update\_plan (::quartus::periph)

The following table displays information for the `periph::update_plan` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::periph</a> on page 240		
<b>Syntax</b>	<code>periph::update_plan [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Applies all changes to constraints and reloads them into Interface Planner. After the platform has been updated with the constraints, placement operations can be performed.		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan blueprint::shutdown project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.22.29. periph::write\_plan (::quartus::periph)

The following table displays information for the periph::write\_plan Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::periph on page 240		
<b>Syntax</b>	periph::write_plan [-h   -help] [-long_help] [-clocks] [-disabled] -filename <filename> [-force] [-other_locations] [-pin_locations]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-clocks	Write out clock assignments	
	-disabled	Write out disabled assignments	
	-filename <filename>	Filename to write to	
	-force	Force the creation of the plan	
	-other_locations	Write out other location assignments	
	-pin_locations	Write out pin location assignments	
<b>Description</b>	Export the floorplan constraints Tcl script		
<b>Example Usage</b>	<pre>project_open onewire_nf blueprint::initialize periph::update_plan periph::place_cells -unplaced_cells periph::check_plan periph::write_plan -filename onewire_blueprint_assignments.tcl project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.23. ::quartus::pfg

The following table displays information for the ::quartus::pfg Tcl package:

<b>Tcl Package and Version</b>	::quartus::pfg 1.0
<b>Description</b>	This package contains the set of Tcl functions for using the Programming File Generator (PFG).
<b>Availability</b>	This package is loaded by default in the following executable: quartus_pfg
<b>Tcl Commands</b>	<a href="#">test (::quartus::pfg) on page 257</a>

### 3.1.23.1. test (::quartus::pfg)

The following table displays information for the test Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::pfg on page 256		
<b>Syntax</b>	test [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Dispaly general information		
<b>Example Usage</b>	test		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24. ::quartus::project

The following table displays information for the ::quartus::project Tcl package:

<b>Tcl Package and Version</b>	::quartus::project 7.0
<b>Description</b>	This package contains the set of Tcl functions for making project-wide assignments. In versions before 4.0 of this package, the full path of the source file assignment was returned when you accessed the assignment through the "get_global_assignment" or "get_all_global_assignments" command. In version 4.0 of this package, the actual value of the source file assignment stored in the Quartus Prime Settings File (.qsf) is returned. To get the resolved full path of the file, use the "resolve_file_path" command. For more information about resolving file names and view an example, type "resolve_file_path -long_help". In version 5.0 of this package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the following commands: get_all_quartus_defaults get_all_global_assignments get_all_instance_assignments get_all_parameters These two new commands simplify the interface to retrieve information about Quartus Prime Settings File (.qsf) and Quartus Prime Default Settings File (.qdf) assignments. In addition, the new "assignment_group" command replaces the deprecated "timegroup" command. In version 6.0, all Tcl commands designed to process Timing Analyzer assignments have been moved to the ::quartus::timing_assignment package.
<b>Availability</b>	This package is loaded by default in the following executables:  hdb_debug gpro_sh quartus_asm quartus_bpps quartus_cdb quartus_design quartus_edt quartus_fit quartus_idb quartus_ipd quartus_ipgenerate quartus_map quartus_sh quartus_si quartus_sim quartus_sta quartus_stp quartus_syn quartus_tlg
<b>Tcl Commands</b>	<a href="#">create_revision (::quartus::project) on page 263</a> <a href="#">delete_revision (::quartus::project) on page 264</a> <a href="#">execute_assignment_batch (::quartus::project) on page 265</a> <a href="#">export_assignments (::quartus::project) on page 265</a> <a href="#">get_all_assignment_names (::quartus::project) on page 266</a> <a href="#">get_all_assignments (::quartus::project) on page 267</a> <a href="#">get_all_global_assignments (::quartus::project) on page 270</a>

*continued...*

```

get_all_instance_assignments (::quartus::project) on page 272
get_all_parameters (::quartus::project) on page 274
get_all_quartus_defaults (::quartus::project) on page 275
get_all_user_option_names (::quartus::project) on page 277
get_assignment_info (::quartus::project) on page 278
get_assignment_name_info (::quartus::project) on page 278
get_current_project (::quartus::project) on page 279
get_current_revision (::quartus::project) on page 279
get_database_version (::quartus::project) on page 280
get_global_assignment (::quartus::project) on page 280
get_instance_assignment (::quartus::project) on page 282
get_location_assignment (::quartus::project) on page 283
get_name_info (::quartus::project) on page 284
get_names (::quartus::project) on page 285
get_parameter (::quartus::project) on page 287
get_project_directory (::quartus::project) on page 288
get_project_revisions (::quartus::project) on page 288
get_top_level_entity (::quartus::project) on page 290
get_user_option (::quartus::project) on page 290
is_database_version_compatible (::quartus::project) on page 291
is_fitter_in_qhd_mode (::quartus::project) on page 291
is_project_open (::quartus::project) on page 292
project_archive (::quartus::project) on page 292
project_clean (::quartus::project) on page 293
project_close (::quartus::project) on page 294
project_exists (::quartus::project) on page 295
project_new (::quartus::project) on page 295
project_open (::quartus::project) on page 296
project_restore (::quartus::project) on page 297
remove_all_global_assignments (::quartus::project) on page 299
remove_all_instance_assignments (::quartus::project) on page 300
remove_all_parameters (::quartus::project) on page 302
resolve_file_path (::quartus::project) on page 304
revision_exists (::quartus::project) on page 305
set_current_revision (::quartus::project) on page 305
set_global_assignment (::quartus::project) on page 306
set_high_effort_fmax_optimization_assignments (::quartus::project) on page 308
set_instance_assignment (::quartus::project) on page 309
set_io_assignment (::quartus::project) on page 311
set_location_assignment (::quartus::project) on page 312
set_parameter (::quartus::project) on page 315
set_power_file_assignment (::quartus::project) on page 317
set_user_option (::quartus::project) on page 320
test_assignment_trait (::quartus::project) on page 321

```

### **3.1.24.1. assignment\_group (::quartus::project)**

The following table displays information for the assignment\_group Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	assignment_group [-h   -help] [-long_help] [-add_exception <name>] [-add_member <name>] [-comment <comment>] [-disable] [-fall] [-get_exceptions] [-get_members] [-overwrite] [-remove] [-remove_exception <name>] [-remove_member <name>] [-rise] [-tag <data>] <group_name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-add_exception <name>	Tcl list of exception names to add
	-add_member <name>	Tcl list of member names to add
	-comment <comment>	Comment
	-disable	Option to disable assignment
	-fall	Option applies to falling edge
	-get_exceptions	Option to get collection of assignment group exceptions
	-get_members	Option to get collection of assignment group members
	-overwrite	Option to overwrite existing assignment group with the same group name

*continued...*

	-remove	Option to remove assignment group	
	-remove_exception <name>	Tcl list of exception names to remove	
	-remove_member <name>	Tcl list of member names to remove	
	-rise	Option applies to rising edge	
	-tag <data>	Option to tag data to this assignment	
	<group_name>	Assignment group name	
<b>Description</b>	<p>Adds, removes, gets members of, or gets exceptions to an assignment group. The "assignment_group" command replaces the deprecated "timegroup" command. An assignment group is a custom group of registers and pins. You can use the "-add_member" option to specify register or pin names you want to include in the assignment group. You can use the "-add_exception" option to specify names you want to exclude from the assignment group. You can specify the names using wildcards, that is, using "?" or "*". For example, to add all registers and pins that start with a "b" except those that start with "b c" to a particular assignment group named "group_b", type:</p> <pre>assignment_group "group_b" -add_member "b*" -add_exception "b c"*</pre> <p>To remove members or exceptions from a assignment group, use the "-remove_member" or "-remove_exception" options respectively. The "-get_members" option returns a collection of members in the assignment group. The "-get_exceptions" option returns a collection of exceptions to the assignment group. Each element of the collection is a list in the following format: { {Assignment group name} {&lt;Assignment name&gt;} {&lt;Register or pin name&gt;} } In lists containing members, &lt;Assignment name&gt; is always "ASSIGNMENT_GROUP_MEMBER". In lists containing exceptions, &lt;Assignment name&gt; is always "ASSIGNMENT_GROUP_EXCEPTION". To disable assignments for an entire group, use the "-disable" option, for example: assignment_group "group_a" -disable To disable a particular assignment group assignment, use the "-disable" option with the "-add_member" or "-add_exception" options, for example: assignment_group "group_a" -add_member "m1" -disable assignment_group "group_a" -add_exception "e1" - disable Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) The "export_assignments" and "project_close" Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</p>		
<b>Example Usage</b>	<pre># Make a location assignment for nodes starting # with "r" except those starting with "r s " # and except those starting with "r t " assignment_group "lg1" -add_member "r*" -add_exception "r s *" assignment_group "lg1" -add_exception "r t *"  set_location_assignment IOBANK_2 -to "lg1"  # Remove the "rl" node from assignment group "lg1" assignment_group "lg1" -remove_member "rl"  # Display the members of assignment group "lg1" foreach_in_collection member [assignment_group "lg1" -get_members] {     foreach { group assignment node } \$member { break }         # Print the name of the member         puts \$node }  # Display the exceptions to assignment group "lg1" foreach_in_collection exception [assignment_group "lg1" -get_exceptions] {     foreach { group assignment node } \$exception { break }         # Print the name of the exception         puts \$node }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.

*continued...*

TCL_ERROR	1	ERROR: Can't set revision: <string>. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Revision does not exist: <string>. Specify a legal revision name using the -<string> option.

### 3.1.24.2. `create_base_clock` (::quartus::project)

The following table displays information for the `create_base_clock` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	<code>create_base_clock [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-duty_cycle &lt;integer&gt;] [-entity &lt;entity&gt;] [-fall] -fmax &lt;fmax&gt; [-no_target] [-rise] [-tag &lt;data&gt;] [-target &lt;name&gt;] [-virtual] &lt;clock_name&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-comment <comment>	Comment
	-disable	Option to disable assignment
	-duty_cycle <integer>	Duty cycle
	-entity <entity>	Entity to which to add clock assignment
	-fall	Option applies to falling edge
	-fmax <fmax>	Clock frequency
	-no_target	Option to not assign clock to node
	-rise	Option applies to rising edge
	-tag <data>	Option to tag data to this assignment
	-target <name>	Clock node name
<b>Description</b>	Creates the base clock. The base clock is an absolute clock. The "-fmax" option can take the format: <floating point time value><time unit> For example, if the fmax is 10.55ns, "10.55" is the <floating point time value> and "ns" is the <time unit>. The following table displays possible time units: Time Unit Description ----- s second(s) ms millisecond(s) us microsecond(s) ns nanosecond(s) ps picosecond(s) fs femtosecond(s) Hz hertz KHz kilohertz MHz megahertz GHz gigahertz If you specify the "-virtual" option, the base clock is not assigned to any node in the timing netlist. You cannot specify the "-virtual" option and the "-target" option at the same time. For entity-specific assignments, use the "-entity" option to force the assignment to specified entity. If you do not specify the "-entity" option, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.	

*continued...*

<b>Example Usage</b>	<pre># Specify a clock named "clk50" with # a 50ns period # The command specifies a CLOCK section # in the active project with the 50ns # specification, and adds a # "clk50 : CLOCK_SETTING=clk50" assignment # to the current entity create_base_clock -fmax 50ns clk50  # Specify the same clk50 to a pin with # a different name (myclkpin) create_base_clock -fmax 50ns -target myclkpin clk50  # Specify the entity name to which the clock # is added, using the -entity option # This is needed if the top-level entity name # is other than that of the project # The following command generates a "top_level" entity. # create_base_clock -fmax 50ns -entity top_level clk50</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: The option -<string> was ignored because it is no longer supported. No action is required.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string> and <string>. Specify only one of the two options.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.3. create\_relative\_clock (::quartus::project)

The following table displays information for the `create_relative_clock` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	<code>create_relative_clock [-h   -help] [-long_help] -base_clock &lt;Base clock&gt; [-comment &lt;comment&gt;] [-disable] [-divide &lt;integer&gt;] [-duty_cycle &lt;integer&gt;] [-entity &lt;entity&gt;] [-fall] [-invert] [-multiply &lt;integer&gt;] [-no_target] [-offset &lt;offset&gt;] [-phase_shift &lt;integer&gt;] [-rise] [-tag &lt;data&gt;] [-target &lt;name&gt;] [-virtual] &lt;clock_name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-base_clock &lt;Base clock&gt;</code>	Base clock name
	<code>-comment &lt;comment&gt;</code>	Comment
	<code>-disable</code>	Option to disable assignment
	<code>-divide &lt;integer&gt;</code>	Base clock division factor
	<code>-duty_cycle &lt;integer&gt;</code>	Duty cycle

*continued...*

	-entity <entity>	Entity to which to add clock assignment	
	-fall	Option applies to falling edge	
	-invert	Option to invert base clock	
	-multiply <integer>	Base clock multiplication factor	
	-no_target	Option to not assign clock to node	
	-offset <offset>	Offset from base clock	
	-phase_shift <integer>	Phase shift from base clock	
	-rise	Option applies to rising edge	
	-tag <data>	Option to tag data to this assignment	
	-target <name>	Clock node name	
	-virtual	Option to specify the clock as a virtual clock	
	<clock_name>	Clock name	
<b>Description</b>	<p>Creates a relative clock that derived from the absolute clock. The "-offset" option can take the format: &lt;floating point time value&gt;&lt;time unit&gt; For example, if the offset is 10.55ns, "10.55" is the &lt;floating point time value&gt; and "ns" is the &lt;time unit&gt;. The following table displays possible time units: Time Unit Description ----- s second(s) ms millisecond(s) us microsecond(s) ns nanosecond(s) ps picosecond(s) fs femtosecond(s) The "-phase_shift" option takes an integer that represents degrees of phase shift from the base clock period. For example, if a base clock has a period of 10ns and clk2 is a relative clock derived from the base clock. A phase shift value of 45 applies a 45 degree phase shift to clk2, producing an offset of 1.25ns from the base clock. For a given relative clock, you may specify a phase shift, an offset, or both. If both are specified, they are additive. If you specify the "-virtual" option, the relative clock is not assigned to any node in the timing netlist. You cannot specify the "-virtual" option and the "-target" option at the same time. For entity-specific assignments, use the "-entity" option to force the assignment to specified entity. If you do not specify the "-entity" option, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</p>		
<b>Example Usage</b>	<pre>## Specify a base clock of 10ns create_base_clock -fmax 10ns clk10  ## Specify a relative clock with 2/3 the period create_relative_clock -base_clock clk10 -multiply 2 -divide 3 clk2_3  ## Specify a relative clock with a phase shift of 45 degrees create_relative_clock -base_clock clk10 -phase_shift 45 clk_45 ## or, equivalently, with an offset of 1.25ns create_relative_clock -base_clock clk10 -offset 1.25ns clk_45  ## Specify the entity name to which the clock ## is added, using the -entity option ## This is needed if the top-level entity name is ## other than that of the project ## The following command generates a "top_level" entity create_relative_clock -base_clock clk10 -entity top_level -multiply 2 -divide 3 clk2_3</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.

*continued...*

TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Options are mutually exclusive: <string> and <string>. Specify only one of the two options.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.4. create\_revision (::quartus::project)

The following table displays information for the `create_revision` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>create_revision [-h   -help] [-long_help] [-based_on &lt;revision_name&gt;] [-copy_results] [-new_rev_type &lt;revision_type&gt;] [-root_partition_qdb_file &lt;qdb_file&gt;] [-set_current] &lt;revision_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-based_on <revision_name>	Revision name on which new revision bases its settings	
	-copy_results	Option to copy results from "based_on" revision	
	-new_rev_type <revision_type>	The type of the newly created revision	
	-root_partition_qdb_file <qdb_file>	The Partition Database (.qdb) file for the root partition	
	-set_current	Option to set new revision as current revision	
	<revision_name>	Revision name	
<b>Description</b>	Creates the specified revision. If the revision is not included in the current project, a new revision is created in the project with default settings. If you specify the "-set_current" option, this command sets the newly created revision as the current revision. If you specify the "-based_on" option, the command creates a new revision in the project based on the settings of the based-on revision specified by the option.		
<b>Example Usage</b>	<pre>## Create a new revision called "tmp" create_revision tmp  ## Create a new revision called "tmp" ## and set it as the current revision create_revision tmp -set_current ## This method is the same as create_revision tmp set_current_revision tmp  ## Create a new revision called "speed_ch" ## with settings based on "chiptrip" ## and set it as the current revision create_revision speed_ch -based_on chiptrip -set_current</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: Revision is already the current revision: <string>. No action is required.
	TCL_ERROR	1	ERROR: Based-on revision is not included in the current project: <string>. Make sure the based-on revision name is spelled correctly and included in the current project.

*continued...*

TCL_ERROR	1	ERROR: Can't create revision because the current project uses the device family: <string>. Change the device family or create the revision in another project that uses a different device family.
TCL_ERROR	1	ERROR: Can't create file: <string>. Make sure you have permission to write to the specified file.
TCL_ERROR	1	ERROR: Unable to create a new 'Persona Implementation' revision based on an existing 'Persona Implementation' revision. Specify another revision type or change the base revision.
TCL_ERROR	1	ERROR: Can't create revision: <string>. Specify a legal revision name.
TCL_ERROR	1	ERROR: Can't remove file: <string>. Make sure the file is not read-only and you have permission to write to the specified file.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Didn't create revision because it is already included in current project: <string>. If you want a new revision, specify a different revision name.

### 3.1.24.5. delete\_revision (::quartus::project)

The following table displays information for the delete\_revision Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	delete_revision [-h   -help] [-long_help] <revision_name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<revision_name>	Revision name	
<b>Description</b>	Deletes the specified revision from the current project. The corresponding <revision name>.qsf file is deleted as well.		
<b>Example Usage</b>	<pre>## Delete the revision called "tmp" delete_revision tmp</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't delete the current revision: <string>. Specify a different revision name.
	TCL_ERROR	1	ERROR: Can't delete revision because it is not included in the current project: <string> . Specify a revision name that is included in the project.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.6. execute\_assignment\_batch (::quartus::project)

The following table displays information for the execute\_assignment\_batch Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257				
<b>Syntax</b>	execute_assignment_batch [-h   -help] [-long_help] <tcl commands>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	<tcl commands>	Tcl list of Tcl commands			
<b>Description</b>	Iterates through the specified Tcl list of Tcl commands and executes each command sequentially in batch mode. In batch mode, Tcl commands that set Quartus Prime Settings File (.qsf) assignments are optimized to prevent them from repeatedly write-locking and write-unlocking the QSF during consecutive calls, thereby slowing down the execution. Currently, only the following commands are supported: assignment_group remove_all_global_assignments remove_all_instance_assignments remove_all_parameters set_global_assignment set_instance_assignment set_io_assignment set_location_assignment set_parameter set_power_file_assignment				
<b>Example Usage</b>	<pre>project_open one_wire set tcl_cmds [list [list set_global_assignment -name FAMILY StratixII] \                    [list set_global_assignment -name DEVICE AUTO] \                    [list set_global_assignment -name TOP_LEVEL_ENTITY one_wire] \                    [list set_global_assignment -name SAVE_DISK_SPACE OFF] \                    [list set_location_assignment PIN_1 -to in1] \                    [list set_instance_assignment -name MULTICYCLE 4 -from in1 -to out1] \                    [list set_parameter -name STYLE FAST]] execute_assignment_batch \$tcl_cmds project_close</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Unsupported Tcl command: <string>. Specify one of the supported Tcl commands listed in the help description for <string> -h.		

### 3.1.24.7. export\_assignments (::quartus::project)

The following table displays information for the export\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	export_assignments [-h   -help] [-long_help] [-reorganize] [-report_write_failure]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-reorganize	Option to reorganize the Quartus Prime Settings File (.qsf)	
	-report_write_failure	Option to report error if write fail	
<b>Description</b>	Exports assignments for the current revision to the Quartus Prime Settings File (.qsf). Assignments created or modified during an open project are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.		

*continued...*

<b>Example Usage</b>	<pre>## The most common use of export_assignments is to ## call it before doing a system call ## to call a compiler command-line executable project_open \$project_name set_global_assignment -name FAMILY Stratix  ## Before calling quartus_map, ## write out the FAMILY assignment export_assignments  ## Now, call quartus_map qexec "[file join \$::quartus(binpath) quartus_map] \$project_name"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't write settings file *.qsf. Make sure the *.qsf file is writeable.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.8. get\_all\_assignment\_names (::quartus::project)

The following table displays information for the `get_all_assignment_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>get_all_assignment_names [-h   -help] [-long_help] [-family &lt;family&gt;] [-module &lt;all map fit tan asm eda drc generic&gt;] [-type &lt;all global instance&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-family &lt;family&gt;</code>	Option to filter based on the specified device family. Defaults to all families.
	<code>-module &lt;all map fit tan asm eda drc generic&gt;</code>	Option to filter based on the specified flow module. Defaults to all.
	<code>-type &lt;all global instance&gt;</code>	Option to filter based on the specified assignment type. Defaults to all.
<b>Description</b>	Returns a filtered output list of all available, matching assignment names. The module option takes one of the following values: Module Description ----- map Analysis and Synthesis assignment names fit Fitter assignment names asm Assembler assignment names eda EDA Netlist Writer assignment names drc Design Assistant assignment names generic Other assignment names not included in any of the above flow modules all All assignment names	
<b>Example Usage</b>	<pre>## Print out all available global assignments foreach i [get_all_assignment_names -type global] {     puts \$i }  ## Print out all available global assignments ## for the Stratix family</pre>	

*continued...*

	<pre> foreach i [get_all_assignment_names -type global -family Stratix] {     puts \$i }  ## Print out all available global assignments ## for the Stratix family required ## by the Analysis and Synthesis module foreach i [get_all_assignment_names -type global -family Stratix -module map] {     puts \$i } </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
	TCL_ERROR	1	ERROR: Illegal flow module: <string>. Specify <string>, <string>, <string>, <string>, <string>, <string>, or <string>.
	TCL_ERROR	1	ERROR: Illegal type: <string>. Specify <string>, <string>, or <string>.
	TCL_ERROR	1	ERROR: Illegal device family: <string>. Specify a legal device family.

### 3.1.24.9. get\_all\_assignments (::quartus::project)

The following table displays information for the `get_all_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<code>get_all_assignments [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] [-from &lt;source&gt;] -name &lt;name&gt; [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;] [-to &lt;destination&gt;] -type &lt;global instance parameter default&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-entity <entity_name>	Entity name
	-fall	Option applies to falling edge
	-from <source>	Source name (string pattern is matched using Tcl string matching)
	-name <name>	Assignment name (string pattern is matched using Tcl string matching)
	-rise	Option applies to rising edge
	-section_id <section id>	Section id
	-tag <data>	Option to tag data to this assignment
	-to <destination>	Destination name (string pattern is matched using Tcl string matching)
<b>Description</b>	Option to specify the type of assignments to return	
	Returns a collection of all matching global, instance, parameter, or default assignment ids. To iterate through each assignment id in this collection, use the Tcl command "foreach_in_collection". To view details for the assignment that is associated with the assignment id, use the Tcl command "get_assignment_info". The "get_all_assignments" command is easier to use than the deprecated	

**continued...**

	<p>commands listed in Table 1. * Table 1. The -type Option Value for -type Option Deprecated Tcl command Description ----- default get_all_quartus_defaults Returns only default assignments. global get_all_global_assignments Returns only global assignments. instance get_all_instance_assignments Returns only instance assignments. parameter get_all_parameters Returns only parameter assignments. The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive. These options can take string patterns containing special characters from the set "*?[][" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format: &lt;bus name&gt;[&lt;bus index&gt;] or &lt;bus name&gt;[*] The &lt;bus name&gt; portion is a string of alphanumeric characters. The &lt;bus index&gt; portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the &lt;bus index&gt; is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows: # To match index 0 of bus "a", type: get_all_assignments -type instance -name LOCATION -to a[0] # To match all indices of bus "a", type: get_all_assignments -type instance -name LOCATION -to a[*] All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type: get_all_assignments -type instance LOCATION -to "a\escape_brackets \\\\[0-2\\\]\escape_brackets \\]" For more information about escaping square brackets, type "escape_brackets -h". This Tcl command reads in the global, instance, and parameter assignments found in the Quartus Prime Settings File (.qsf) and reads in the default assignments found inside the Quartus Prime Default Settings File (.qdf). If you tagged data by making assignments with the -tag option, then the information can be searched using the -tag option. Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one clock section each containing its own set of clock assignments. To uniquely identify sections of this type, use the -section_id option. For entity-specific assignments, use the "-entity" option to retrieve assignments from a specific entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.</p>
<b>Example Usage</b>	<pre> ## View all the timing requirements using wildcards ## to match TSU_REQUIREMENT, TCO_REQUIREMENT, ## and others. foreach_in_collection asgn_id [get_all_assignments -type instance -name *_REQUIREMENT] {     set from      [get_assignment_info \$asgn_id -from]     set to        [get_assignment_info \$asgn_id -to]     set name      [get_assignment_info \$asgn_id -name]     set value     [get_assignment_info \$asgn_id -value]     set entity    [get_assignment_info \$asgn_id -entity]     set sid       [get_assignment_info \$asgn_id -section_id]     set tag       [get_assignment_info \$asgn_id -tag]      puts "\$entity: \$name (\$from -&gt; \$to) = \$value" }  ## View all global assignments foreach_in_collection asgn_id [get_all_assignments -type global -name *] {     set name      [get_assignment_info \$asgn_id -name]     set value     [get_assignment_info \$asgn_id -value]     set entity    [get_assignment_info \$asgn_id -entity]     set sid       [get_assignment_info \$asgn_id -section_id]     set tag       [get_assignment_info \$asgn_id -tag]      puts "\$entity: \$name = \$value" }  ## View all project-wide default parameter values foreach_in_collection asgn_id [get_all_assignments -type parameter -name *] {     set name      [get_assignment_info \$asgn_id -name]     set value     [get_assignment_info \$asgn_id -value]     set tag       [get_assignment_info \$asgn_id -tag]      puts "\$name = \$value" }  ## View all entity-specific parameter values foreach_in_collection asgn_id [get_all_assignments -type parameter -name * -to *] {     set dest      [get_assignment_info \$asgn_id -to]     set name      [get_assignment_info \$asgn_id -name]     set value     [get_assignment_info \$asgn_id -value]     set tag       [get_assignment_info \$asgn_id -tag]      puts "\$name (-&gt; \$dest) = \$value" }  ## View all default assignments foreach_in_collection asgn_id [get_all_assignments -type default -name * -to *] { </pre>

*continued...*

		<pre> set name  [get_assignment_info \$asgn_id -name] set value  [get_assignment_info \$asgn_id -value] puts "\$name = \$value" } </pre>	
Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_OK	0		INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
TCL_ERROR	1		ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.
TCL_ERROR	1		ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
TCL_ERROR	1		ERROR: Can't find section information for assignment. Specify a different assignment name.
TCL_ERROR	1		ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1		ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1		ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1		ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1		ERROR: Illegal assignment type: <string>. Specify <string>, <string>, <string>, or <string>.
TCL_ERROR	1		ERROR: Illegal option <string>. The specified option is illegal for <string> assignments.
TCL_ERROR	1		ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1		ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1		ERROR: You must open a project before you can use this command.
TCL_ERROR	1		ERROR: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1		ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1		ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1		ERROR: An unknown error has occurred.

### 3.1.24.10. get\_all\_global\_assignments (::quartus::project)

The following table displays information for the `get_all_global_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>get_all_global_assignments [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-rise] [-section_id &lt;section_id&gt;] [-tag &lt;data&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-entity &lt;entity_name&gt;</code>	Entity to which assignment belongs
	<code>-fall</code>	Option applies to falling edge
	<code>-name &lt;name&gt;</code>	Assignment name (string pattern is matched using Tcl string matching)
	<code>-rise</code>	Option applies to rising edge
	<code>-section_id &lt;section_id&gt;</code>	Section id
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment
<b>Description</b>	<p>Returns a filtered output collection of all matching global assignment values. To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help". In version 5.0 of the ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_global_assignments" command. These two new commands simplify the interface to retrieve information about Quartus Prime Settings File (.qsf) assignments. The "get_all_global_assignments" command is still supported for backward compatibility. The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\\[]" as the value. The value is matched using Tcl string matching. This Tcl command reads the global assignments found in the Quartus Prime Settings File (.qsf). This Tcl command filters the assignment data in the .qsf and outputs the data based on the values given by the "-name" option. Each element of the collection is a list with the following format: { {&lt;Section Id&gt;} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;} {&lt;Entity name&gt;} {&lt;Tag data&gt;} } Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a &lt;Section Id&gt; is used. &lt;Section Id&gt; can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a &lt;Section Id&gt; and the associated &lt;Section Id&gt; description: Section Id Description ----- CHIP Same as revision name LOGICLOCK_REGION A unique name EDA_TOOL_SETTINGS A unique name CLIQUE A unique name BREAKPOINT A unique name CLOCK A unique name AUTO_INSERT_SLD_NODE_ENTITY A unique name If you tagged data by making assignments with the -tag option, then the information will be displayed in the &lt;Tag data&gt; field. For entity-specific assignments, use the "-entity" option to retrieve the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.</p>	
<b>Example Usage</b>	<pre>## Print out all the registered source files ## using the foreach_in_collection method set_file_asgn_col [get_all_global_assignments -name SOURCE_FILE] foreach_in_collection file_asgn \$file_asgn_col {     ## Each element in the collection has the following     ## format: {} {SOURCE_FILE} {&lt;file_name&gt;} {} {}     puts [lindex \$file_asgn 2]  }  ## Print out all global assignments set_asgn_col [get_all_global_assignments -name *]  foreach_in_collection asgn \$asgn_col {     ## Each element in the collection has the following     ## format: { {} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;} {&lt;Entity name&gt;} {&lt;Tag data&gt;} }     set name [lindex \$asgn 1]     set value [lindex \$asgn 2]</pre>	

*continued...*

			<pre> set entity [lindex \$asgn 3] set tag    [lindex \$asgn 4] puts "\$entity: \$name = \$value" } </pre>
Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_OK		0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
TCL_ERROR	1		ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
TCL_ERROR	1		ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
TCL_ERROR	1		ERROR: Can't find section information for assignment. Specify a different assignment name.
TCL_ERROR	1		ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1		ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1		ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1		ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1		ERROR: Option -<string> for <string> assignment is illegal. Specify a legal option or remove the option.
TCL_ERROR	1		ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1		ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1		ERROR: You must open a project before you can use this command.
TCL_ERROR	1		ERROR: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1		ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1		ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1		ERROR: An unknown error has occurred.

### 3.1.24.11. get\_all\_instance\_assignments (::quartus::project)

The following table displays information for the `get_all_instance_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>get_all_instance_assignments [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] [-from &lt;source&gt;] -name &lt;name&gt; [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-entity &lt;entity_name&gt;</code>	Entity to which assignment belongs
	<code>-fall</code>	Option applies to falling edge
	<code>-from &lt;source&gt;</code>	Source of assignment (string pattern is matched using Tcl string matching)
	<code>-name &lt;name&gt;</code>	Assignment name (string pattern is matched using Tcl string matching)
	<code>-rise</code>	Option applies to rising edge
	<code>-section_id &lt;section id&gt;</code>	Section id
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment
	<code>-to &lt;destination&gt;</code>	Destination of assignment (string pattern is matched using Tcl string matching)
<b>Description</b>	<p>Returns a filtered output collection of all matching instance assignment values. To access each element of this output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help". In version 5.0 of the ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_instance_assignments" command. These two new commands simplify the interface to retrieve information about Quartus Prime Settings File (.qsf) assignments. The "get_all_instance_assignments" command is still supported for backward compatibility. The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive. These options can take string patterns containing special characters from the set "?[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format: &lt;bus name&gt;[&lt;bus index&gt;] or &lt;bus name&gt;[*]. The &lt;bus name&gt; portion is a string of alphanumeric characters. The &lt;bus index&gt; portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the &lt;bus index&gt; is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows: # To match index 0 of bus "a", type:  <code>get_all_instance_assignments -name LOCATION -to a[0]</code> # To match all indices of bus "a", type:  <code>get_all_instance_assignments -name LOCATION -to a[*]</code> All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type: <code>get_all_instance_assignments -name LOCATION -to "a[escape_brackets \[\] \[0-2\][escape_brackets \]]"</code> For more information about escaping square brackets, type "escape_brackets -h". This Tcl command reads in the instance assignments found in the Quartus Prime Settings File (.qsf). The command filters the assignments data found in the .qsf and outputs the data based on the values specified by the "-name", "-from", and "-to" options. Each element of the collection is a list with the following format: { {&lt;Section Id&gt;} {&lt;Source&gt;} {&lt;Destination&gt;} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;} {&lt;Entity name&gt;} {&lt;Tag data&gt;} } Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a &lt;Section Id&gt; is used. &lt;Section Id&gt; can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a &lt;Section Id&gt; and the associated &lt;Section Id&gt; description:            Section Id Description ----- CHIP Same as revision            name LOGICLOCK_REGION A unique name EDA_TOOL_SETTINGS A unique name CLIQUE A unique name BREAKPOINT A unique name CLOCK A unique name AUTO_INSERT_SLD_NODE_ENTITY A unique name If you tagged data by making assignments with the -tag option, then the information</p>	

*continued...*

	<p>will be displayed in the &lt;Tag data&gt; field. For entity-specific assignments, use the "-entity" option to retrieve the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.</p>																																							
<b>Example Usage</b>	<pre>## Print out all the timing requirements ## using the foreach_in_collection method. ## Use wildcards to catch TSU_REQUIREMENT, TCO_REQUIREMENT, ## and others. set asgn_col [get_all_instance_assignments -name *_REQUIREMENT] foreach_in_collection asgn \$asgn_col {     ## Each element in the collection has the following     ## format: { {} {&lt;Source&gt;} {&lt;Destination&gt;} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;}     {&lt;Entity name&gt;} {&lt;Tag data&gt;} }     set from [lindex \$asgn 1]     set to [lindex \$asgn 2]     set name [lindex \$asgn 3]     set value [lindex \$asgn 4]     set entity [lindex \$asgn 5]     set tag [lindex \$asgn 6]     puts "\$entity: \$name (\$from -&gt; \$to) = \$value" } ## Get all the location assignments with ## the destination bus name "timeo". set bus_name "timeo" set location_asgns [get_all_instance_assignments -name LOCATION -to \$bus_name[*]]</pre>																																							
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Assignment is not an instance assignment: &lt;string&gt; -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Can't find section information for assignment. Specify a different assignment name.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Can't find active revision name. Make sure there is an open, active revision name.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Entity does not exist or uses illegal name characters: &lt;string&gt;. Specify a legal entity name.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Can't run Tcl command while a process is in progress: &lt;string&gt;. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Value &lt;string&gt; for the -&lt;string&gt; option is illegal. Specify a legal value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Options cannot be specified together: -&lt;string&gt;, -&lt;string&gt; and -&lt;string&gt;. Specify only one or two of the three options.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Missing destination for assignment. Specify the destination for the assignment.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: You must open a project before you can use this command.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Ignored assignment: &lt;string&gt;. The assignment is no longer supported.</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.	TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.	TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.	TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.	TCL_ERROR	1	ERROR: You must open a project before you can use this command.	TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.
Code Name	Code	String Return																																						
TCL_OK	0	INFO: Operation successful																																						
TCL_ERROR	1	ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.																																						
TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.																																						
TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.																																						
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.																																						
TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.																																						
TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.																																						
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.																																						
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.																																						
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.																																						
TCL_ERROR	1	ERROR: You must open a project before you can use this command.																																						
TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.																																						

*continued...*

TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.12. `get_all_parameters` (::quartus::project)

The following table displays information for the `get_all_parameters` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<code>get_all_parameters [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-rise] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-entity <entity_name>	Entity to which parameter belongs
	-fall	Option applies to falling edge
	-name <name>	Parameter name (string pattern is matched using Tcl string matching)
	-rise	Option applies to rising edge
	-tag <data>	Option to tag data to this assignment
	-to <destination>	Destination of the parameter (string pattern is matched using Tcl string matching)
<b>Description</b>	<p>Returns a filtered output collection of all matching parameter values. To access each element of this output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help". In version 5.0 of ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_parameters" command. These two new commands simplify the interface to retrieve information about Quartus Prime Settings File (.qsf) assignments. The "get_all_parameters" command is still supported for backward compatibility. The "-name" option is not case sensitive. The "-to" option is case sensitive. If the "-to" argument is specified, the function returns the parameter values for the current entity. The values are retrieved from the PARAMETERS section of the entity. Otherwise, the function returns the project-wide default parameter values obtained from the DEFAULT_PARAMETERS section. This Tcl command filters the parameter data found in the Quartus Prime Settings File (.qsf) and outputs the data based on the values specified by the "-name" and "-to" options. These options can take string patterns containing special characters from the set "*?\\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format: &lt;bus name&gt;[&lt;bus index&gt;] or &lt;bus name&gt;[*]. The &lt;bus name&gt; portion is a string of alphanumeric characters. The &lt;bus index&gt; portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the &lt;bus index&gt; is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows: # To match index 0 of bus "a", type: <code>get_all_parameters -name * -to a[0]</code> # To match all indices of bus "a", type: <code>get_all_parameters -name * -to a[*]</code>. All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type: <code>get_all_parameters -name * -to a[escape_brackets \\[]\\[0-2\\]\\[escape_brackets \\]]</code>. For more information about escaping square brackets, type "escape_brackets -h". Each element of the collection is a list with the following format: { {&lt;Destination&gt;} {&lt;Parameter name&gt;} {&lt;Parameter value&gt;} {&lt;Entity name&gt;} {&lt;Tag data&gt;} }. If you tagged data by making assignments with the -tag option, then the information will be displayed in the &lt;Tag data&gt; field. Use the "-entity" option to retrieve the parameter values from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.</p>	

*continued...*

<b>Example Usage</b>	<pre>## Display all project-wide default parameter values set parameter_col [get_all_parameters -name *]  foreach_in_collection parameter \$parameter_col {      ## Each element in the collection has the following     ## format: { {} {&lt;Parameter name&gt;} {&lt;Parameter value&gt;} {} {} }     set name [lindex \$parameter 1]     set value [lindex \$parameter 2]      ## Now, display the content of the parameter     puts "Parameter Name (\$name)"     puts "Parameter Value (\$value)"  }  ## Display all entity-specific parameter values foreach_in_collection parameter [get_all_parameters -name * -to *] {      ## Each element in the collection has the following     ## format: { {Destination} {&lt;Parameter name&gt;} {&lt;Parameter value&gt;} {} {} }     set dest [lindex \$parameter 0]     set name [lindex \$parameter 1]     set value [lindex \$parameter 2]      ## Now, display the content of the parameter     puts "Destination (\$dest)"     puts "Parameter Name (\$name)"     puts "Parameter Value (\$value)"  }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Illegal default parameter: <string>. Specify a legal default parameter name.
	TCL_ERROR	1	ERROR: Illegal parameter: <string>. Specify a legal parameter name.
	TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.13. get\_all\_quartus\_defaults (::quartus::project)

The following table displays information for the `get_all_quartus_defaults` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::project</code> on page 257	
<b>Syntax</b>	<code>get_all_quartus_defaults [-h   -help] [-long_help] [-fall] [-name &lt;name&gt;] [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-fall</code>	Option applies to falling edge
	<code>-name &lt;name&gt;</code>	Assignment name (string pattern is matched using Tcl string matching)
	<code>-rise</code>	Option applies to rising edge
	<code>-section_id &lt;section id&gt;</code>	Section id

*continued...*

	<pre>-tag &lt;data&gt;</pre>	Option to tag data to this assignment	
<b>Description</b>	<p>Returns a filtered output collection of all matching default assignment values. To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help". In version 5.0 of ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_quartus_defaults" command. These two new commands simplify the interface to retrieve information about Quartus Prime Settings File (.qsf) assignments. The "get_all_quartus_defaults" command is still supported for backward compatibility. The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\\[]" as the value. The value is matched using Tcl string matching. This Tcl command reads in the default assignments found inside the Quartus Prime Default Settings File (.qdf). It filters the assignments data found inside the .qdf and outputs the data based on the values specified by the "-name" option. Each element of the collection is a list with the following format: { {&lt;Section Id&gt;} {&lt;Assignment name&gt;} {&lt;Assignment value&gt;} } Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a &lt;Section Id&gt; is used. &lt;Section Id&gt; can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a &lt;Section Id&gt; and the associated &lt;Section Id&gt; description: Section Id Description ----- ----- CHIP Same as revision name LOGICLOCK_REGION A unique name EDA_TOOL_SETTINGS A unique name CLIQUE A unique name BREAKPOINT A unique name CLOCK A unique name AUTO_INSERT_SLD_NODE_ENTITY A unique name</p>		
<b>Example Usage</b>	<pre>## Print out all the default assignments using ## the foreach_in_collection method  set default_asgns_col [get_all_quartus_defaults] foreach_in_collection default \$default_asgns_col {     set sect_id [lindex \$default 0]     set name [lindex \$default 1]     set value [lindex \$default 2]      ## Now, display the content of the assignment     puts "Section ID (\$sect_id)"     puts "Assignment Name (\$name)"     puts "Assignment Value (\$value)" }  ## Using wildcards set default_asgns_col [get_all_quartus_defaults -name *] foreach_in_collection default \$default_asgns_col {     set sect_id [lindex \$default 0]     set name [lindex \$default 1]     set value [lindex \$default 2]      ## Now, display the content of the assignment     puts "Section ID (\$sect_id)"     puts "Assignment Name (\$name)"     puts "Assignment Value (\$value)" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.

*continued...*

TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.14. `get_all_user_option_names` (::quartus::project)

The following table displays information for the `get_all_user_option_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257				
<b>Syntax</b>	<code>get_all_user_option_names [-h   -help] [-long_help] [-name &lt;name&gt; ]</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-name <name>	User option name (string pattern is matched using Tcl string matching)			
<b>Description</b>	Returns a filtered output list of all available, matching user option names. If the "-name" option is not specified, all available user option names are returned. Otherwise, only the matching user option names are returned. The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\\[]" as the value. The value is matched using Tcl string matching.				
<b>Example Usage</b>	<pre>## Print out all available user option names foreach i [get_all_user_option_names] {     puts \$i }  ## Display all user option names that contain ## the word "talkback" and also display the ## value for each of the user option names foreach i [get_all_user_option_names -name *talkback*] {     set name \$i     set value [get_user_option -name \$i]     puts "\$name = \$value" }</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.24.15. get\_assignment\_info (::quartus::project)

The following table displays information for the `get_assignment_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>get_assignment_info [-h   -help] [-long_help] [-comments] [-entity] [-from] [-get_tcl_command] [-name] [-section_id] [-tag] [-to] [-value] &lt;asgn_id&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-comments	Option to get the assignment comment	
	-entity	Option to get the assignment entity	
	-from	Option to get the assignment source	
	-get_tcl_command	Option to get the tcl command that sets the assignment	
	-name	Option to get the assignment name	
	-section_id	Option to get the assignment section id	
	-tag	Option to get the assignment tag	
	-to	Option to get the assignment destination	
	-value	Option to get the assignment value	
	<asgn_id>	Assignment id	
<b>Description</b>	Returns information for the assignment id based on the specified option. The assignment id is obtained from the "get_all_assignments" Tcl command.		
<b>Example Usage</b>	<pre>## View all the instance assignments foreach_in_collection asgn_id [get_all_assignments -type instance -name *] {     set from  [get_assignment_info \$asgn_id -from]     set to    [get_assignment_info \$asgn_id -to]     set name  [get_assignment_info \$asgn_id -name]     set value [get_assignment_info \$asgn_id -value]     set entity [get_assignment_info \$asgn_id -entity]     set tag   [get_assignment_info \$asgn_id -tag]      puts "\$entity: \$name (\$from -&gt; \$to) = \$value" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Illegal assignment id: <string>. Specify a legal assignment id that was retrieved from the Tcl command <code>get_all_assignments</code> .

### 3.1.24.16. get\_assignment\_name\_info (::quartus::project)

The following table displays information for the `get_assignment_name_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>get_assignment_name_info [-h   -help] [-long_help] &lt;name&gt;</code>		
	<i>continued...</i>		

<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<name>	Assignment name	
<b>Description</b>	Returns information for the specified assignment name.		
<b>Example Usage</b>	<pre>## View information for all assignment names foreach name [get_all_assignment_names] {     puts [get_assignment_name_info \$name] }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
	TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.

### 3.1.24.17. `get_current_project` (::quartus::project)

The following table displays information for the `get_current_project` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_current_project [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns the name of the current project.		
<b>Example Usage</b>	<pre># Get the current name for # the currently open project "chiptrip" project_open chiptrip set project_name [get_current_project] project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.18. `get_current_revision` (::quartus::project)

The following table displays information for the `get_current_revision` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_current_revision [-h   -help] [-long_help] [ &lt;project_name&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<project_name>	Project name	

*continued...*

<b>Description</b>	Returns the name of the current revision for the specified project. If the project name is not specified, the current project name is used.		
<b>Example Usage</b>	<pre># Get the current revision name for # the currently open project "chiptrip" project_open chiptrip set revision_name [get_current_revision] project_close  # Get the current revision name for # a project that is not currently open set revision_name [get_current_revision chiptrip]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Project does not exist or has illegal name characters: <string>. Specify a legal project name.
	TCL_ERROR	1	ERROR: Project name was not specified or open project does not exist. Open an existing project or specify the project name.

### 3.1.24.19. get\_database\_version (::quartus::project)

The following table displays information for the get\_database\_version Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	get_database_version [-h   -help] [-long_help] <project_name>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	<project_name>		Project name
<b>Description</b>	Return the project's database version. This is the version of software that the database was created.		
<b>Example Usage</b>	<pre>## Print message on database version set db_version [get_database_version chiptrip] post_message "The database is created from \"\$db_version\""</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.20. get\_global\_assignment (::quartus::project)

The following table displays information for the get\_global\_assignment Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	get_global_assignment [-h   -help] [-long_help] [-entity <entity_name>] [-fall] [-front] -name <name> [-rise] [-section_id <section_id>] [-tag <data>]		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-entity <entity_name>		Entity to which assignment belongs

*continued...*

	-fall	Option applies to falling edge	
	-front	Option to return the first assignment if there is more than one assignment found	
	-name <name>	Assignment name	
	-rise	Option applies to rising edge	
	-section_id <section id>	Section id	
	-tag <data>	Option to tag data to this assignment	
<b>Description</b>	Returns the value of the global assignment. The "-name" option is not case sensitive. For entity-specific assignments, use the "-entity" option to retrieve the assignment from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.		
<b>Example Usage</b>	<pre>## Get the value of the FAMILY assignment get_global_assignment -name FAMILY</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
	TCL_ERROR	1	ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
	TCL_ERROR	1	ERROR: Option -<string> for <string> assignment is illegal. Specify a legal option or remove the option.
	TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
	TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.
	TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.

*continued...*

TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.
TCL_ERROR	1	ERROR: Assignment <string> has multiple values. Use the <string> command to get all values or use the <string> -front command to get the first value.

### 3.1.24.21. `get_instance_assignment` (::quartus::project)

The following table displays information for the `get_instance_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>get_instance_assignment [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] [-from &lt;source&gt;] [-front] -name &lt;name&gt; [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-entity <entity_name>	Entity to which assignment belongs	
	-fall	Option applies to falling edge	
	-from <source>	Source of assignment	
	-front	Option to return the first assignment if there is more than one assignment found	
	-name <name>	Assignment name	
	-rise	Option applies to rising edge	
	-section_id <section id>	Section id	
	-tag <data>	Option to tag data to this assignment	
<b>Description</b>	Returns the value of the instance assignment. The "-name" option is not case sensitive. The "-entity", "-to", and "-from" options are case sensitive. For entity-specific assignments, use the "-entity" option to retrieve the assignment from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.		
	<pre>## Get the TSU_REQUIREMENT from mypin to any register set value [get_instance_assignment -from "mypin" -to * -name TSU_REQUIREMENT] puts "TSU_REQUIREMENT(mypin-&gt;*) = \$value"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.
	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.

*continued...*

TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.
TCL_ERROR	1	ERROR: Assignment <string> has multiple values. Use the <string> command to get all values or use the <string> -front command to get the first value.

### 3.1.24.22. `get_location_assignment` (::quartus::project)

The following table displays information for the `get_location_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_location_assignment [-h   -help] [-long_help] [-fall] [-rise] [-tag &lt;data&gt;] -to &lt;destination&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fall	Option applies to falling edge	
	-rise	Option applies to rising edge	
	-tag <data>	Option to tag data to this assignment	
	-to <destination>	Destination of assignment	
<b>Description</b>	Returns the value of a location assignment. The "-chip" option is not case sensitive. The "-to" option is case sensitive.		
<b>Example Usage</b>	<code>get_location_assignment -to dst</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.23. `get_name_info` (::quartus::project)

The following table displays information for the `get_name_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257																																										
<b>Syntax</b>	<code>get_name_info [-h   -help] [-long_help] [-info &lt;parent_name_id base_name entity_name entity_definition instance_name full_path short_full_path node_type creator signalapii file_location library children parameters&gt;] [-observable_type &lt;all pre_synthesis post_synthesis post_fitter post_asm stp_pre_synthesis&gt;] &lt;name_id&gt;</code>																																										
<b>Arguments</b>	<code>-h   -help</code>	Short help																																									
	<code>-long_help</code>	Long help with examples and possible return values																																									
	<code>-info &lt;parent_name_id base_name entity_name entity_definition instance_name full_path short_full_path node_type creator signalapii file_location library children parameters&gt;</code>	Option to specify the type of information to display.																																									
	<code>-observable_type &lt;all pre_synthesis post_synthesis post_fitter post_asm stp_pre_synthesis&gt;</code>	Option to specify the observable type of the name ID																																									
	<code>&lt;name_id&gt;</code>	Option to specify the node name ID																																									
	<p><b>Description</b></p> <p>Displays the specified type of information for the specified node name id. Type "get_names -long_help" to view how to get a collection of node name IDs. If the "-observable_type" option is not specified, the default value is "all". The specified observable type must have the same observable type as specified in the "get_names" Tcl command which returned the currently specified node name id. The value for "-observable_type" option can be one of the following:</p> <table> <tr> <td>-----</td> <td>all</td> <td>Use post-Fitter information. If it is not available, post-synthesis information is used. Otherwise, pre-synthesis information is used if it exists.</td> </tr> <tr> <td>pre_synthesis</td> <td>Use pre-synthesis information.</td> <td>post_synthesis</td> <td>Use post-synthesis information.</td> </tr> <tr> <td>post_fitter</td> <td>Use post-Fitter information.</td> <td>post_asm</td> <td>Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family.</td> </tr> <tr> <td>stp_pre_synthesis</td> <td>Use Signal Tap pre-synthesis information.</td> <td></td> <td></td> </tr> </table> <p>The info type for the "-info" option can be one of the following:</p> <table> <tr> <td>-----</td> <td>parent_name_id</td> <td>The name id for the node's parent.</td> </tr> <tr> <td>base_name</td> <td>The node name, which consists of an entity name and/or an instance name separated by a colon if necessary.</td> </tr> <tr> <td>entity_name</td> <td>The entity name.</td> </tr> <tr> <td>entity_definition</td> <td>The entity definition.</td> </tr> <tr> <td>instance_name</td> <td>The instance name.</td> </tr> <tr> <td>full_path</td> <td>The full hierarchy path name, which consists of entity name(s) and/or the instance name(s). This path name excludes the current focus entity. If there is nothing shown, the name id is the current focus entity's name id.</td> </tr> <tr> <td>short_full_path</td> <td>The short full hierarchy path name, which consists of the instance name(s). This path name excludes the current focus entity. If nothing is shown, the name id is the current focus entity's name id.</td> </tr> <tr> <td>node_type</td> <td>The node type, which can be one of the types supported by "get_names", namely, "input", "output", "bidirectional", "register", "combinational", "hierarchy", "memory", or "bus". If "pin" type was specified for "get_names" command, the node type shown here is expanded to be "input", "output", or "bidirectional". Node type value of "qsf" indicates name originates from qsf settings file.</td> </tr> <tr> <td>creator</td> <td>The creator of the node, which is either "user_entered" or "compiler_generated".</td> </tr> <tr> <td>signalapii</td> <td>If this node can be connected to a Signal Tap embedded logic analyzer, 1 is shown. Otherwise, 0 is shown.</td> </tr> <tr> <td>file_location</td> <td>The source file location. For example, the source file location for the entity chiptrip is "chiptrip.v". To get the full path to the source file, use the command "resolve_file_path" which exists only in version 4.0 or later of ::quartus::project package.</td> </tr> <tr> <td>library</td> <td>Library associated with the instance name.</td> </tr> <tr> <td>children</td> <td>Collection of all the children names of the</td> </tr> </table>		-----	all	Use post-Fitter information. If it is not available, post-synthesis information is used. Otherwise, pre-synthesis information is used if it exists.	pre_synthesis	Use pre-synthesis information.	post_synthesis	Use post-synthesis information.	post_fitter	Use post-Fitter information.	post_asm	Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family.	stp_pre_synthesis	Use Signal Tap pre-synthesis information.			-----	parent_name_id	The name id for the node's parent.	base_name	The node name, which consists of an entity name and/or an instance name separated by a colon if necessary.	entity_name	The entity name.	entity_definition	The entity definition.	instance_name	The instance name.	full_path	The full hierarchy path name, which consists of entity name(s) and/or the instance name(s). This path name excludes the current focus entity. If there is nothing shown, the name id is the current focus entity's name id.	short_full_path	The short full hierarchy path name, which consists of the instance name(s). This path name excludes the current focus entity. If nothing is shown, the name id is the current focus entity's name id.	node_type	The node type, which can be one of the types supported by "get_names", namely, "input", "output", "bidirectional", "register", "combinational", "hierarchy", "memory", or "bus". If "pin" type was specified for "get_names" command, the node type shown here is expanded to be "input", "output", or "bidirectional". Node type value of "qsf" indicates name originates from qsf settings file.	creator	The creator of the node, which is either "user_entered" or "compiler_generated".	signalapii	If this node can be connected to a Signal Tap embedded logic analyzer, 1 is shown. Otherwise, 0 is shown.	file_location	The source file location. For example, the source file location for the entity chiptrip is "chiptrip.v". To get the full path to the source file, use the command "resolve_file_path" which exists only in version 4.0 or later of ::quartus::project package.	library	Library associated with the instance name.	children
-----	all	Use post-Fitter information. If it is not available, post-synthesis information is used. Otherwise, pre-synthesis information is used if it exists.																																									
pre_synthesis	Use pre-synthesis information.	post_synthesis	Use post-synthesis information.																																								
post_fitter	Use post-Fitter information.	post_asm	Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family.																																								
stp_pre_synthesis	Use Signal Tap pre-synthesis information.																																										
-----	parent_name_id	The name id for the node's parent.																																									
base_name	The node name, which consists of an entity name and/or an instance name separated by a colon if necessary.																																										
entity_name	The entity name.																																										
entity_definition	The entity definition.																																										
instance_name	The instance name.																																										
full_path	The full hierarchy path name, which consists of entity name(s) and/or the instance name(s). This path name excludes the current focus entity. If there is nothing shown, the name id is the current focus entity's name id.																																										
short_full_path	The short full hierarchy path name, which consists of the instance name(s). This path name excludes the current focus entity. If nothing is shown, the name id is the current focus entity's name id.																																										
node_type	The node type, which can be one of the types supported by "get_names", namely, "input", "output", "bidirectional", "register", "combinational", "hierarchy", "memory", or "bus". If "pin" type was specified for "get_names" command, the node type shown here is expanded to be "input", "output", or "bidirectional". Node type value of "qsf" indicates name originates from qsf settings file.																																										
creator	The creator of the node, which is either "user_entered" or "compiler_generated".																																										
signalapii	If this node can be connected to a Signal Tap embedded logic analyzer, 1 is shown. Otherwise, 0 is shown.																																										
file_location	The source file location. For example, the source file location for the entity chiptrip is "chiptrip.v". To get the full path to the source file, use the command "resolve_file_path" which exists only in version 4.0 or later of ::quartus::project package.																																										
library	Library associated with the instance name.																																										
children	Collection of all the children names of the																																										

*continued...*

	specified name. The children will include all the names in the specified hierarchy. parameters Collection of parameters associated with the name. Each element of the collection is a triplet that contains the name, value and the type of the parameter.		
<b>Example Usage</b>	<pre># Get the name id of the current focus entity set current_focus_entity_id [get_top_level_entity]  # The full path name of the current focus entity # is empty because the full path excludes the # current focus entity set msg "Full path of the current focus entity =&gt; (" append msg [get_name_info -info full_path \$current_focus_entity_id] append msg ")" puts \$msg puts ""  # Get the node type of the current focus entity # The node type should be a hierarchy type set msg "Node type of the current focus entity =&gt; (" append msg [get_name_info -info node_type \$current_focus_entity_id] append msg ")" puts \$msg</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Compiler database does not exist for revision name: <string>. At the minimum, run Analysis and Synthesis (quartus_map) with the specified revision name before using this Tcl command.
	TCL_ERROR	1	ERROR: Illegal info type: <string>. Specify parent_name_id, base_name, entity_name, instance_name, full_path, short_full_path, node_type, creator, or signaltapii.
	TCL_ERROR	1	ERROR: Illegal name id: <string>. Specify a name id that exists in a compiled Quartus Prime project.
	TCL_ERROR	1	ERROR: Invalid name id for top level entity: <string>. Specify a valid top level entity id value. In Quartus Prime Pro, get_name_info query through integer value is only supported for get_top_level_entity Tcl command. For other names query, please refer to get_names Tcl command.
	TCL_ERROR	1	ERROR: Invalid name id: <string>. Specify an integer greater than or equal to zero.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Post-Assembler compiler database does not exist for revision name: <string>. Run Assembler (quartus_asm) with the specified revision name before using this Tcl command.

### 3.1.24.24. get\_names (::quartus::project)

The following table displays information for the get\_names Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<pre>get_names [-h   -help] [-long_help] [-entity &lt;wildcard&gt;] -filter &lt;wildcard&gt; [-library &lt;wildcard&gt;] [-node_type &lt;all comb reg pin input output bidir hierarchy mem bus qsf state_machine assigned unassigned all_reg partition virtual&gt;] [-observable_type &lt;all pre_synthesis post_synthesis post_fitter post_asm stp_pre_synthesis&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
<i>continued...</i>		

	<ul style="list-style-type: none"> <li>-long_help Long help with examples and possible return values</li> <li>-entity &lt;wildcard&gt; Option to specify the entity to get names from hierarchies instantiated by the entity</li> <li>-filter &lt;wildcard&gt; Option to specify the node's full path name and/or wildcard character(s)</li> <li>-library &lt;wildcard&gt; Option to specify the containing library</li> <li>-node_type &lt;all comb reg pin input output bidir hierarchy mem bus qsf state_machine assigned unassigned all_reg partition virtual&gt; Option to filter based on the specified node type.</li> <li>-observable_type &lt;all pre_synthesis post_synthesis post_fitter post_asm stp_pre_synthesis&gt; Option to filter based on the specified observable type</li> </ul>
<b>Description</b>	<p>Returns a filtered output collection of all matching node name IDs found in a compiled Quartus Prime project. To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "get_names -long_help" or "foreach_in_collection -long_help". If the "-node_type" option is not specified, the default value is "all". Similarly, if the "-observable_type" option is not specified, the default value is "all". The node type "pin" includes "input", "output", "bidir", "assinged" and "unassigned". The node type "qsf" include names from qsf settings file. The node type "all" includes all node types. The node type "all_reg" includes all node types and register post-fitting. The value for "-observable_type" option can be one of the following:</p> <ul style="list-style-type: none"> <li>----- all Use post-Fitter information. If it is not available, post-Synthesis information is used. Otherwise, pre-synthesis information is used if it exists.</li> <li>pre_synthesis Use pre-synthesis information.</li> <li>post_synthesis Use post-synthesis information.</li> <li>post_fitter Use post-Fitter information.</li> <li>post_asm Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family.</li> <li>stp_pre_synthesis Use Signal Tap pre-synthesis information.</li> </ul>
<b>Example Usage</b>	<pre># Search for a single post-Fitter pin with the name accel and # make assignments set accel_name_id [get_names -filter accel -node_type pin -observable_type post_fitter] foreach_in_collection name_id \$accel_name_id {     # Get the full path name of the node     set target [get_name_info -info full_path \$name_id]      # Set multicycle assignment     set_multicycle_assignment -to \$target 2      # Set location assignment     set_location_assignment -to \$target Pin_E22 } # Search for nodes of any post-Fitter node type with name length &lt;= 5 # The default node type is "all" set name_ids [get_names -filter ???? -observable_type post_fitter] foreach_in_collection name_id \$name_ids {     # Print the name id     puts \$name_id      # Print the node type     puts [get_name_info -info node_type \$name_id]      # Print the full path (which excludes the current     # focus entity from the path)     puts [get_name_info -info full_path \$name_id] } # Search for nodes of any post-Fitter node type that end in "eed". # The default node type is "all" set name_ids [get_names -filter *eed -observable_type post_fitter] foreach_in_collection name_id \$name_ids {     # Print the name id     puts \$name_id      # Print the node type     puts [get_name_info -info node_type \$name_id]      # Print the full path (which excludes the current </pre>

continued...

	<pre> # focus entity from the path) puts [get_name_info -info full_path \$name_id] } </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Get names cannot return <string> because the name was found in a partition that's not the root partition. Refine your get_names search pattern to exclude child partitions
	TCL_ERROR	1	ERROR: Compiler database does not exist for revision name: <string>. At the minimum, run Analysis and Synthesis (quartus_map) with the specified revision name before using this Tcl command.
	TCL_ERROR	1	ERROR: Illegal node type: <string>. Specify all, comb, reg, pin, hierarchy, or bus.
	TCL_ERROR	1	ERROR: Illegal observable type: <string>. Specify all, pre_synthesis, post_synthesis, or post_fitter.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Post-Assembler compiler database does not exist for revision name: <string>. Run Assembler (quartus_asm) with the specified revision name before using this Tcl command.

### 3.1.24.25. `get_parameter` (::quartus::project)

The following table displays information for the `get_parameter` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>get_parameter [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-rise] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-entity &lt;entity_name&gt;</code>	Entity to which parameter belongs
	<code>-fall</code>	Option applies to falling edge
	<code>-name &lt;name&gt;</code>	Parameter name
	<code>-rise</code>	Option applies to rising edge
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment
	<code>-to &lt;destination&gt;</code>	Destination of parameter
<b>Description</b>	<p>Returns the value of the parameter. The "-name" option is not case sensitive. The "-to" option is case sensitive. If the "-to" argument is specified, the function returns the parameter value for the current entity. The value is retrieved from the PARAMETERS section of the entity. Otherwise, the function returns the project-wide default parameter value obtained from the DEFAULT_PARAMETERS section. Use the "-entity" option to retrieve the parameter from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.</p>	

*continued...*

<b>Example Usage</b>	<pre>## Get project-wide, default parameter value get_parameter -name WIDTH  ## Get entity-specific parameter value get_parameter -name instl -to SIZE</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Illegal default parameter: <string>. Specify a legal default parameter name.
	TCL_ERROR	1	ERROR: Illegal parameter: <string>. Specify a legal parameter name.
	TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.26. `get_project_directory` (::quartus::project)

The following table displays information for the `get_project_directory` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_project_directory [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
<b>Description</b>	Returns the project directory for currently open project.		
<b>Example Usage</b>	<pre>project_open one_wire # Print the current project directory puts [get_project_directory] project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.27. `get_project_revisions` (::quartus::project)

The following table displays information for the `get_project_revisions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_project_revisions [-h   -help] [-long_help] [ &lt;project_name&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
<i>continued...</i>			

	-long_help	Long help with examples and possible return values	
	<project_name>	Project name	
<b>Description</b>	Returns a list of revisions included in the specified project. If the project name is not specified, the current project name is used by default. The first element in the list of revisions is the current revision and is the same as the return value for the "get_current_revision" command.		
<b>Example Usage</b>	<pre># Set the device family assignment to Stratix # for all revisions project_open chiptrip set original_revision [get_current_revision]  foreach revision [get_project_revisions] {     puts "\$revision"     set_current_revision \$revision     set_global_assignment -name FAMILY Stratix     export_assignments }  set_current_revision \$original_revision project_close  # Open the project with the first available revision # and set the device family assignment to Stratix set revision [lindex [get_project_revisions chiptrip] 0] open_project -revision \$revision chiptrip set_global_assignment -name FAMILY Stratix project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Project does not exist or has illegal name characters: <string>. Specify a legal project name.
	TCL_ERROR	1	ERROR: Project name was not specified or open project does not exist. Open an existing project or specify the project name.

### 3.1.24.28. get\_project\_settings (::quartus::project)

The following table displays information for the `get_project_settings` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0		
<b>Syntax</b>	<code>get_project_settings [-h   -help] [-long_help] [-cmp] [-sim] [-swb]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cmp	Revision name	
	-sim	Revision name	
	-swb	Revision name	
<b>Description</b>	Returns the name of the current revision regardless of which option was specified.		
<b>Example Usage</b>	<pre>set revision_name [get_project_settings -cmp] set revision_name [get_project_settings -sim] set revision_name [get_project_settings -swb]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.29. `get_top_level_entity` (::quartus::project)

The following table displays information for the `get_top_level_entity` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_top_level_entity [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns the name id for the current focus entity.		
<b>Example Usage</b>	<pre># Get the name id of the current focus entity set current_focus_entity_id [get_top_level_entity]  # Print out the entity name of the current focus entity set msg "Entity name of the current focus entity =&gt; ("  append msg [get_name_info -info entity_name \$current_focus_entity_id] append msg ")" puts "" puts \$msg</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Compiler database does not exist for revision name: <string>. At the minimum, run Analysis and Synthesis (quartus_map) with the specified revision name before using this Tcl command.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.30. `get_user_option` (::quartus::project)

The following table displays information for the `get_user_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	<code>get_user_option [-h   -help] [-long_help] -name &lt;name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-name &lt;name&gt;</code>	User option name	
<b>Description</b>	Returns the user option value for the name specified by the "-name" option. To get a list of all available user option names, use the "get_all_user_option_names" command.		

*continued...*

<b>Example Usage</b>	<pre>## Get the value for the user option ## "TALKBACK_ENABLED" set value [get_user_option -name TALKBACK_ENABLED] puts "TALKBACK_ENABLED = \$value"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.31. is\_database\_version\_compatible (::quartus::project)

The following table displays information for the is\_database\_version\_compatible Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	is_database_version_compatible [-h   -help] [-long_help] <project_name>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	<project_name>		Project name
<b>Description</b>	Checks whether a project's database is compatible with current version of the software. Returns 1, if the database is compatible; returns 0, otherwise. Mainly this check is for databases that are restored from another version of the software		
<b>Example Usage</b>	<pre>## Check project database version compatibility if [is_database_version_compatible chiptrip] {     # proceed if compatible }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.32. is\_fitter\_in\_qhd\_mode (::quartus::project)

The following table displays information for the is\_fitter\_in\_qhd\_mode Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	is_fitter_in_qhd_mode [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
<b>Description</b>	Returns true if we're running in QHD mode		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.33. **is\_project\_open** (::quartus::project)

The following table displays information for the `is_project_open` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>is_project_open [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Checks whether a project is currently open. Returns 1, if a project is currently open; returns 0, otherwise.		
<b>Example Usage</b>	<pre>## Close the project if open if [is_project_open] {     project_close }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.34. **project\_archive** (::quartus::project)

The following table displays information for the `project_archive` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>project_archive [-h   -help] [-long_help] [-all_revisions] [-include_libraries] [-include_outputs] [-overwrite] [-use_file_set &lt;file_set&gt;] [-version_compatible_database] &lt;archive_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-all_revisions</code>	Option to archive all revisions	
	<code>-include_libraries</code>	Option to include related system libraries	
	<code>-include_outputs</code>	Option to include output files in archive	
	<code>-overwrite</code>	Option to overwrite any currently existing archive file	
	<code>-use_file_set &lt;file_set&gt;</code>	Option to create the archive using the specified file set	
	<code>-version_compatible_database</code>	Option to include version-compatible database if supported	
	<code>&lt;archive_name&gt;</code>	Archive file name	
<b>Description</b>	Archives an open project and its related files into a Quartus Prime Archive File (.qar). The description of operations is as follows: Option Description ----- ----- use_file_set Creates the archive using the specified file set. By default, the 'basic' file set is used. For more information about file sets, type: quartus_sh --archive -list_file_sets all_revisions Archives all revisions. overwrite Overwrites existing archive file. include_outputs Includes output files in archive. include_libraries Includes related Megafunction and IP library files. version_compatible_database Includes version-compatible database if supported.		
<b>Example Usage</b>	<pre>## Default mode: Archive current revisions without output files or libraries project_archive chiptrip.qar  ## Archive all revisions without output files or libraries project_archive chiptrip.qar -all_revisions  ## Archive current revision with version-compatible database if supported</pre>		

*continued...*

	<pre>project_archive chiptrip.qar -version_compatible_database ## Same as first one, but overwrite any existing archive file project_archive chiptrip.qar -overwrite ## Include output files and libraries project_archive chiptrip.qar -include_outputs -include_libraries</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Error(s) found while archiving the project. See error message(s) for details.
	TCL_ERROR	1	ERROR: Project archive failed. Some files could not be processed. Refer to the Quartus Prime Archive Log File (<archive_name>.qarlog).
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.35. project\_clean (::quartus::project)

The following table displays information for the project\_clean Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	project_clean [-h   -help] [-long_help] [-current_version] [-revision <revision_name> ]		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-current_version		Only clean project files for the current Quartus version.
	-revision <revision_name>		Revision to clean (if omitted, all revisions of the open project are cleaned). Revision name can contain wildcards (i.e., '*').
<b>Description</b>	Cleans database and compiler-generated output for the specified revision (or all revisions if no revision is specified). Cleaning revisions removes database and other files generated by the Quartus Prime software, including report and programming files. This command closes the currently open project before cleaning. Specifically, this cleans all of the following files/folders (for all matching revisions): <revision>.*.rpt <revision>.*.rpt.htm <revision>.*.rpt.htm_files <revision>.*.msf <revision>.*.smsg <revision>.*.summary <revision>.jdi <revision>.pin <revision>.pof <revision>.sof <revision>.done <revision>.sld (all revision files in the qdb and persona directories)		
<b>Example Usage</b>	<pre>project_clean -revision "foo"    # Cleans revision "foo" project_clean -revision "foo*"   # Cleans all revisions starting with "foo" (e.g., "foo", "foobar") project_clean                         # Cleans all revisions in project</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Failed when attempting to remove database files in <string> for revision <string>.
	TCL_ERROR	1	ERROR: Failed when attempting to remove file <string> for revision <string>.

*continued...*

TCL_ERROR	1	ERROR: Couldn't delete file or folder named '<string>'.
TCL_ERROR	1	ERROR: Couldn't open configuration settings...the settings object is currently locked.
TCL_ERROR	1	ERROR: Can't access revision '<string>'. The revision may not exist, or there may be an error in the revision settings.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.36. project\_close (::quartus::project)

The following table displays information for the project\_close Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	project_close [-h   -help] [-long_help] [-dont_export_assignments]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-dont_export_assignments	Do not export assignments to file	
<b>Description</b>	Closes an open project. The assignments created or modified during an open project are committed to the Quartus Prime Settings File (.qsf) during a "project_close", unless you use the "-dont_export_assignments" option.		
<b>Example Usage</b>	<pre>## Close the project if open if [is_project_open] {     project_close } ## Close the project if open ## and do not export the assignments if [is_project_open] {     project_close -dont_export_assignments }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Failed when attempting to write assignments back to QSF.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.37. project\_exists (::quartus::project)

The following table displays information for the `project_exists` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>project_exists [-h   -help] [-long_help] &lt;project_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>&lt;project_name&gt;</code>	Project name	
<b>Description</b>	Checks whether a project exists. Returns 1, if a project exists; returns 0, otherwise.		
<b>Example Usage</b>	<pre>## Create project if one does not exist. ## Open existing project otherwise. if [project_exists chiptrip] {     project_open chiptrip } else {     project_new chiptrip }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.24.38. project\_new (::quartus::project)

The following table displays information for the `project_new` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>project_new [-h   -help] [-long_help] [-family &lt;family&gt;] [-overwrite] [-part &lt;part&gt;] [-revision &lt;revision_name&gt;] &lt;project_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-family &lt;family&gt;</code>	Family name	
	<code>-overwrite</code>	Option to overwrite existing project and revision	
	<code>-part &lt;part&gt;</code>	Part name	
	<code>-revision &lt;revision_name&gt;</code>	Revision name	
	<code>&lt;project_name&gt;</code>	Project name	
<b>Description</b>	Creates and opens a new project with the specified project name. If the "-revision" option is not specified, the project name is used to create the revision. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless " <code>-dont_export_assignments</code> " is specified) These two Tcl commands reside in the <code>::quartus::project</code> package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the <code>::quartus::flow</code> Tcl package) automatically call "export_assignments" before they run command-line executables.		
<b>Example Usage</b>	<pre>## Create project "chiptrip" and revision "chiptrip" project_new chiptrip  ## Create project "chiptrip" and revision "auto_max" project_new -revision auto_max chiptrip  ## Create project "chiptrip" and revision "chiptrip"</pre>		

*continued...*

	<pre>## Overwrite any Quartus Prime Settings File (.qsf) if it exists project_new chiptrip -overwrite  ## Create project "chiptrip" and revision "chiptrip" ## Set the FAMILY assignment to Stratix project_new chiptrip -family Stratix</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The -<string> option must also be used when you use the -<string> option. Specify both options.
	TCL_ERROR	1	ERROR: Can't create project: <string>. Specify a legal project name.
	TCL_ERROR	1	ERROR: Can't create revision: <string>. Specify a legal revision name using the -<string> option.
	TCL_ERROR	1	ERROR: Can't create revision: <string>. Specify a legal revision name.
	TCL_ERROR	1	ERROR: Can't create settings files for project: <string>. Make sure the .psf, .csf, and .ssf files are writeable.
	TCL_ERROR	1	ERROR: Can't open project: <string>
	TCL_ERROR	1	ERROR: Can't remove Quartus Prime Settings File: <string>. Make sure the file is writeable.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.

### 3.1.24.39. `project_open (::quartus::project)`

The following table displays information for the `project_open` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>project_open [-h   -help] [-long_help] [-current_revision] [-force] [-revision &lt;revision_name&gt;] &lt;project_name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-current_revision</code>	Option to open the current revision automatically
	<code>-force</code>	Option to open the project and overwrite the compilation database if the database version is incompatible.
	<code>-revision &lt;revision_name&gt;</code>	Revision name
	<code>&lt;project_name&gt;</code>	Project name
<b>Description</b>	Opens an existing project. To create a new project, use the <code>project_new</code> command. If the <code>-revision</code> option is not specified, the project name is specified as the revision name. The <code>project_open</code> command gives an error when the compilation database version is not compatible with the current version of Quartus Prime software. You may specify the " <code>-force</code> " option to avoid the error and overwrite the database.	
<i>continued...</i>		

<b>Example Usage</b>	<pre>## Open project "chiptrip" and revision "chiptrip" project_open chiptrip  ## Open project "chiptrip" and revision "auto_max" project_open -revision auto_max chiptrip  ## Get the current revision before opening ## the project with the current revision set project_name chiptrip set current_revision [get_current_revision \$project_name] project_open -revision \$current_revision \$project_name puts [get_global_assignment -name FAMILY] project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: Project is already open: <string>
	TCL_ERROR	1	ERROR: Can't open project: <string>. First close the currently open project: <string>.
	TCL_ERROR	1	ERROR: Can't open project: <string>
	TCL_ERROR	1	ERROR: Can't set revision: <string>. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: Cannot open project: <string>. The project is not compatible with the installed version of the Quartus Prime software. Opening the project will overwrite the old project database. If you wish to overwrite the old project database, make sure to specify the -<string> option.
	TCL_ERROR	1	ERROR: Can't open revision: <string> (project: <string>). The revision is not compatible with the installed version of the Quartus Prime software. Opening the revision will overwrite the old revision database. If you wish to overwrite the old revision database, make sure to specify the -<string> option.
	TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.

### 3.1.24.40. project\_restore (::quartus::project)

The following table displays information for the project\_restore Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	project_restore [-h   -help] [-long_help] [-destination <directory>] [-overwrite] [-update_included_file_info] <archive_file>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-destination <directory>	Directory where restored files are placed

*continued...*

	-overwrite	Option to overwrite files in destination directory	
	-update_included_file_info	Option to update included file information	
	<archive_file>	Archive file name	
<b>Description</b>	Restores a Quartus Prime Archive File (.qar) that contains the project and its related files. By default, the archive is restored into the current directory. Use the "-destination" option to restore the files into a new directory. By default, the command fails if the archive already contains files in the destination directory. Use the "-overwrite" option to overwrite any existing files in the destination directory.		
<b>Example Usage</b>	<pre>## Restore archive and expand files into current directory project_restore chiptrip.qar ## or project_restore chiptrip.qar -destination  ## Restore archive. Expand files into current directory, ## but overwrite any existing files in "." project_restore chiptrip.qar -destination . -overwrite  ## Restore project into a "restored" subdirectory project_restore chiptrip.qar -destination "restored" -overwrite</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: Error(s) found while restoring the archive. See error message(s) for details.

### 3.1.24.41. project\_settings\_exist (::quartus::project)

The following table displays information for the project\_settings\_exist Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	project_settings_exist [-h   -help] [-long_help] [-cmp <revision_name>] [-sim <revision_name>] [-swb <revision_name>]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-cmp <revision_name>	Revision name
	-sim <revision_name>	Revision name
	-swb <revision_name>	Revision name
<b>Description</b>	Checks whether the specified revision exists. Returns 1, if revision exists; returns 0, otherwise.	
<b>Example Usage</b>	<pre>## Check if auto_max.qsf exists project_settings_exist -cmp auto_max  ## Check if auto_max.qsf exists project_settings_exist -sim auto_max  ## Check if auto_max.qsf exists project_settings_exist -swb auto_max  ## Check if auto_max.qsf exists project_settings_exist -swb auto_max</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.42. remove\_all\_global\_assignments (::quartus::project)

The following table displays information for the `remove_all_global_assignments` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257																
<b>Syntax</b>	<code>remove_all_global_assignments [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-rise] [-section_id &lt;section_id&gt;] [-tag &lt;data&gt;]</code>																
<b>Arguments</b>	-h   -help	Short help															
	-long_help	Long help with examples and possible return values															
	-entity <entity_name>	Entity to which assignment belongs															
	-fall	Option applies to falling edge															
	-name <name>	Assignment name (string pattern is matched using Tcl string matching)															
	-rise	Option applies to rising edge															
	-section_id <section_id>	Section id															
	-tag <data>	Option to tag data to this assignment															
<b>Description</b>	<p>Removes all matching global assignments. The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?[]{}" as the value. The value is matched using Tcl string matching. This Tcl command reads the global assignments found in the Quartus Prime Settings File (.qsf). This Tcl command filters the assignments data found in the .qsf and removes the data based on the values specified by the "-name" option. Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a &lt;Section Id&gt; is used. &lt;Section Id&gt; can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a &lt;Section Id&gt; and the associated &lt;Section Id&gt; description:</p> <table border="0"> <tr> <td>Section Id Description</td> <td>-----</td> <td>CHIP Same as revision</td> </tr> <tr> <td>name LOGICLOCK_REGION</td> <td>A unique name</td> <td>EDA_TOOL_SETTINGS A unique name</td> </tr> <tr> <td>BREAKPOINT</td> <td>A unique name</td> <td>CLIQUE A unique name</td> </tr> <tr> <td>CLOCK</td> <td>A unique name</td> <td>AUTO_INSERT_SLD_NODE_ENTITY A unique name</td> </tr> <tr> <td>For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments removed by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</td> <td></td> </tr> </table>			Section Id Description	-----	CHIP Same as revision	name LOGICLOCK_REGION	A unique name	EDA_TOOL_SETTINGS A unique name	BREAKPOINT	A unique name	CLIQUE A unique name	CLOCK	A unique name	AUTO_INSERT_SLD_NODE_ENTITY A unique name	For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments removed by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.	
Section Id Description	-----	CHIP Same as revision															
name LOGICLOCK_REGION	A unique name	EDA_TOOL_SETTINGS A unique name															
BREAKPOINT	A unique name	CLIQUE A unique name															
CLOCK	A unique name	AUTO_INSERT_SLD_NODE_ENTITY A unique name															
For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments removed by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.																	
<b>Example Usage</b>	<pre>## Remove all the registered source files remove_all_global_assignments -name SOURCE_FILE  # Using wildcards remove_all_global_assignments -name SOURCE*</pre>																
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>														

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_OK	0	INFO: <string> global assignment(s) were removed
TCL_OK	0	WARNING: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1	ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Option -<string> for <string> assignment is illegal. Specify a legal option or remove the option.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.43. remove\_all\_instance\_assignments (::quartus::project)

The following table displays information for the remove\_all\_instance\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<pre>remove_all_instance_assignments [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] [-from &lt;source&gt;] -name &lt;name&gt; [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values

*continued...*

	-entity <entity_name>	Entity to which assignment belongs	
	-fall	Option applies to falling edge	
	-from <source>	Source of the assignment (string pattern is matched using Tcl string matching)	
	-name <name>	Assignment name (string pattern is matched using Tcl string matching)	
	-rise	Option applies to rising edge	
	-section_id <section id>	Section id	
	-tag <data>	Option to tag data to this assignment	
	-to <destination>	Destination of the assignment (string pattern is matched using Tcl string matching)	
<b>Description</b>	<p>Removes all matching instance assignment values. The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive. These options can take string patterns containing special characters from the set "*?\\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format: &lt;bus name&gt;[&lt;bus index&gt;] or &lt;bus name&gt;[*]. The &lt;bus name&gt; portion is a string of alphanumeric characters. The &lt;bus index&gt; portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the &lt;bus index&gt; is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows: # To match index 0 of bus "a", type: remove_all_instance_assignments -name LOCATION -to a[0] # To match all indices of bus "a", type: remove_all_instance_assignments -name LOCATION -to a[*] All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:</p> <pre>remove_all_instance_assignments -name LOCATION -to "a[escape_brackets \[\]\[0-2\] [escape_brackets \]]]"</pre> <p>For more information about escaping square brackets, type "escape_brackets -h". This Tcl command reads the instance assignments found in the Quartus Prime Settings File (.qsf) and removes this data based on the values specified by the "-name", "-from", and "-to" options.</p> <p>Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a &lt;Section Id&gt; is used. &lt;Section Id&gt; can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a &lt;Section Id&gt; and the associated &lt;Section Id&gt; description:</p> <ul style="list-style-type: none"> <li>----- CHIP Same as revision name</li> <li>----- LOGICLOCK_REGION A unique name</li> <li>----- EDA_TOOL_SETTINGS A unique name</li> <li>----- CLIQUE A unique name</li> <li>----- BREAKPOINT A unique name</li> <li>----- CLOCK A unique name</li> <li>----- AUTO_INSERT_SLD_NODE_ENTITY A unique name</li> <li>----- For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. Assignments removed by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</li> </ul>		
<b>Example Usage</b>	<pre>## Remove all the timing requirements ## Use wildcards to catch TSU_REQUIREMENT, TCO_REQUIREMENT, ## and others remove_all_instance_assignments -name *_REQUIREMENT  ## Remove all the location assignments with ## the destination bus name "timeo". set bus_name "timeo" remove_all_instance_assignments -name LOCATION -to \$bus_name[*]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: <string> instance assignment(s) were removed

*continued...*

TCL_OK	0	WARNING: Ignored assignment: <string>. The assignment is no longer supported.
TCL_ERROR	1	ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.
TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.44. remove\_all\_parameters (::quartus::project)

The following table displays information for the `remove_all_parameters` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257	
<b>Syntax</b>	<code>remove_all_parameters [-h   -help] [-long_help] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-rise] [-tag &lt;data&gt;] [-to &lt;destination&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-entity <entity_name>	Entity to which parameter belongs
	-fall	Option applies to falling edge
	-name <name>	Parameter name (string pattern is matched using Tcl string matching)
	-rise	Option applies to rising edge

*continued...*

	-tag <data>	Option to tag data to this assignment	
	-to <destination>	Destination of the parameter (string pattern is matched using Tcl string matching)	
<b>Description</b>	<p>Removes all matching parameters. The "-name" option is not case sensitive. The "-to" option is case sensitive. If the "-to" argument is specified, the function removes the parameters from the current entity. The parameters are removed from the PARAMETERS section of the entity. Otherwise, the function removes the project-wide default parameters obtained from the DEFAULT_PARAMETERS section. This Tcl command filters the parameter data found in the Quartus Prime Settings File (.qsf) and removes the data based on the values specified by the "-name" and "-to" options. These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format: &lt;bus name&gt;[&lt;bus index&gt;] or &lt;bus name&gt;[*]. The &lt;bus name&gt; portion is a string of alphanumeric characters. The &lt;bus index&gt; portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the &lt;bus index&gt; is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows: # To match index 0 of bus "a", type:  remove_all_parameters -name * -to a[0] # To match all indices of bus "a", type:  remove_all_parameters -name * -to a[*]  All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type: remove_all_parameters -name * -to "a[escape_brackets \[]\[\[0-2\]\]\[escape_brackets \]]"  For more information about escaping square brackets, type "escape_brackets -h". Use the "-entity" option to remove the parameters from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used. The parameters removed by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</p>		
<b>Example Usage</b>	<pre>## The following 3 examples remove project-wide, ## default parameter values remove_all_parameters -name WIDTH remove_all_parameters -name *ID* remove_all_parameters -name *</pre> <pre>## The following 3 examples remove entity-specific ## parameter values remove_all_parameters -name inst1 -to SIZE remove_all_parameters -name inst1 -to *IZ* remove_all_parameters -name inst1 -to *</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: <string> parameter(s) were removed
	TCL_OK	0	INFO: Removed parameter: <string>
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: Parameter does not exist and cannot be removed: <string>. Specify an existing parameter name.

*continued...*

TCL_ERROR	1	ERROR: Illegal default parameter: <string>. Specify a legal default parameter name.
TCL_ERROR	1	ERROR: Illegal parameter: <string>. Specify a legal parameter name.
TCL_ERROR	1	ERROR: An unknown error has occurred.

### 3.1.24.45. resolve\_file\_path (::quartus::project)

The following table displays information for the `resolve_file_path` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>resolve_file_path [-h   -help] [-long_help] &lt;file_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>&lt;file_name&gt;</code>	Option to specify the file name	
<b>Description</b>	Returns the resolved full path of the specified file name. If the file does not exist, the original file name is returned. The Quartus Prime software resolves relative paths by searching for the file in the following directories in the following order: 1) Project directory, which is the directory where the Quartus Prime Settings File (.qsf) is found. 2) Project database directory, which is the "db" directory found under the project directory. 3) Project library directories, which are the directories containing the user-specified libraries that are used only by the current project. 4) User library directories, which are the directories containing the user-specified libraries that are used by all Quartus Prime projects. 5) Quartus Prime library directory, which is the directory containing Quartus Prime libraries.		
<b>Example Usage</b>	<pre>project_new chiptrip -overwrite # Set one Verilog source file assignment set_global_assignment -name VERILOG_FILE chiptrip.v  # Display the resolved full path of the Verilog # source file assignment set filename [get_global_assignment -name VERILOG_FILE] set resolvedFullPath [resolve_file_path \$filename]  puts "Full Path: \$resolvedFullPath"  # Set more Verilog source file assignments set_global_assignment -name VERILOG_FILE auto_max.v set_global_assignment -name VERILOG_FILE speed_ch.v set_global_assignment -name VERILOG_FILE tick_cnt.v set_global_assignment -name VERILOG_FILE time_cnt.v  # Display the resolved full path of all the Verilog # source file assignments set file_asgns [get_all_global_assignments -name VERILOG_FILE] foreach_in_collection file_asgn \$file_asgns {     ## Each element in the collection has the following     ## format: {} {VERILOG_FILE} {&lt;file_name&gt;}      set filename [lindex \$file_asgn 2]     set resolvedFullPath [resolve_file_path \$filename]      puts "Full Path: \$resolvedFullPath" } project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.46. revision\_exists (::quartus::project)

The following table displays information for the `revision_exists` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>revision_exists [-h   -help] [-long_help] [-project &lt;project_name&gt;] &lt;revision_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-project &lt;project_name&gt;</code>	Project name	
	<code>&lt;revision_name&gt;</code>	Revision name	
<b>Description</b>	Checks whether the revision exists for the specified project or currently open project. Returns 1, if the revision exists; returns 0, otherwise.		
<b>Example Usage</b>	<pre>## Check if the specified revision exists ## in the specified project if [revision_exists -ARG(project) chiptrip speed_ch] {     puts "Revision exists" } else {     puts "Revision does not exist" }  ## Create revision for the currently open ## project if it does not exist ## Set the current revision otherwise project_open chiptrip if [revision_exists speed_ch] {     set_current_revision speed_ch } else {     create_revision speed_ch } project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Project does not exist or has illegal name characters: <string>. Specify a legal project name.
	TCL_ERROR	1	ERROR: Project name was not specified or open project does not exist. Open an existing project or specify the project name.

### 3.1.24.47. set\_current\_revision (::quartus::project)

The following table displays information for the `set_current_revision` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>set_current_revision [-h   -help] [-long_help] [-force] &lt;revision_name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-force</code>	Option to open the revision and overwrite the compilation database if the database version is incompatible.	
	<code>&lt;revision_name&gt;</code>	Revision name	

*continued...*

<b>Description</b>	Sets the specified revision name as the current revision. All assignments created or modified during an open project will also be saved to the Quartus Prime Settings File (.qsf). In 8.1 or later versions of Quartus Prime software, set_current_revision gives an error when the compilation database version is not compatible with the current version of Quartus Prime software. You may specify the "-force" option to avoid the error and overwrite the database.		
<b>Example Usage</b>	<pre>## Sets "auto_max" as the current revision set_current_revision auto_max</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: Revision is already the current revision: <string>. No action is required.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Revision file does not exist: <string>.qsf. Use delete_revision to delete the revision from the current project. Then use create_revision to create the revision and its .qsf before setting <string> as the current revision.
	TCL_ERROR	1	ERROR: Revision is not included in the current project: <string> . Use the create_revision command to create the revision.

### 3.1.24.48. `set_global_assignment` (::quartus::project)

The following table displays information for the `set_global_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<code>set_global_assignment [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-remove] [-rise] [-section_id &lt;section id&gt;] [-tag &lt;data&gt;] [ &lt;value&gt; ]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-comment <comment>	Comment
	-disable	Option to disable assignment
	-entity <entity_name>	Entity to which to add assignment
	-fall	Option applies to falling edge
	-name <name>	Assignment name
	-remove	Option to remove assignment
	-rise	Option applies to rising edge
	-section_id <section id>	Section id
	-tag <data>	Option to tag data to this assignment
	<value>	Assignment value
<b>Description</b>	Sets or removes a global assignment. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands (from the ::quartus::project Tcl package): 1) export_assignments 2) project_close (unless -dont_export_assignments is specified as an argument to project_close) You must save	

*continued...*

	<p>assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands <code>execute_flow</code> and <code>execute_module</code> (from the <code>::quartus::flow</code> Tcl package) call "export_assignments" before they run command-line executables. For entity-specific assignments, use the <code>-entity</code> option to force the assignment to specified entity. If the <code>-entity</code> option is not specified, the value for the <code>FOCUS_ENTITY_NAME</code> assignment is used. If the <code>FOCUS_ENTITY_NAME</code> value is not found, the revision name is used. If the Quartus Prime Settings File contains a <code>USER_LIBRARIES</code> assignment and you call <code>set_global_assignment</code> to set a <code>SEARCH_PATH</code> or <code>USER_LIBRARIES</code> assignment, the existing <code>USER_LIBRARIES</code> assignment expands into one or more <code>SEARCH_PATH</code> assignments. Note that values that begin with a dash ("") should be enclosed in a backslash followed by a quote. In the following example, <code>-O2</code> is enclosed by <code>\\" at the beginning and the end.</code></p> <pre>set_global_assignment -name ARM_CPP_COMMAND_LINE \\"-O2\\"</pre>		
<b>Example Usage</b>	<pre>## Specify Stratix as the family to use when compiling set_global_assignment -name FAMILY Stratix  ## If the family name has empty spaces, use quotes set_global_assignment -name FAMILY "Stratix GX"  ## or remove any empty space set_global_assignment -name FAMILY StratixGX</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
	TCL_OK	0	WARNING: Ignored assignment: <string>. The assignment is no longer supported.
	TCL_ERROR	1	ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: The assignment <string> is from Design Template and can't be changed/removed.
	TCL_ERROR	1	ERROR: File name <string> exceeds maximum of <string> characters. Specify a file name with fewer characters.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
	TCL_ERROR	1	ERROR: Option -<string> for <string> assignment is illegal. Specify a legal option or remove the option.
	TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
	TCL_ERROR	1	ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.

*continued...*

TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: The -<string> option is not required but was specified with the value: <string>. Delete the option.
TCL_ERROR	1	ERROR: The -<string> option is required but was not specified. Specify the required option.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.
TCL_ERROR	1	ERROR: Assignment <string> cannot be removed -- it has multiple values. Specify one value to remove or use the <string> command to remove all values for the assignment.
TCL_ERROR	1	ERROR: Missing <<string>> for <string> assignment. Specify the required <string>.

### 3.1.24.49. set\_high\_effort\_fmax\_optimization\_assignments (::quartus::project)

The following table displays information for the set\_high\_effort\_fmax\_optimization\_assignments Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257		
<b>Syntax</b>	set_high_effort_fmax_optimization_assignments [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Sets assignments that together implement the high-effort fmax optimization flow. This Tcl command only sets the assignments but does not run a compilation.		
<b>Example Usage</b>	<pre>## Open the project project_open \$project_name  ## Set assignments that implement the high-effort ## fmax optimization flow set_high_effort_fmax_optimization_assignments</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Set global assignment <string> to <string>.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.50. set\_instance\_assignment (::quartus::project)

The following table displays information for the `set_instance_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::project</code> on page 257		
<b>Syntax</b>	<code>set_instance_assignment [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-entity &lt;entity_name&gt;] [-fall] [-from &lt;source&gt;] -name &lt;name&gt; [-remove] [-rise] [-section_id &lt;section_id&gt;] [-tag &lt;data&gt;] [-to &lt;destination&gt;] [&lt;value&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-comment &lt;comment&gt;</code>	Comment	
	<code>-disable</code>	Option to disable assignment	
	<code>-entity &lt;entity_name&gt;</code>	Entity to which to add assignment	
	<code>-fall</code>	Option applies to falling edge	
	<code>-from &lt;source&gt;</code>	Source of assignment	
	<code>-name &lt;name&gt;</code>	Assignment name	
	<code>-remove</code>	Option to remove assignment	
	<code>-rise</code>	Option applies to rising edge	
	<code>-section_id &lt;section_id&gt;</code>	Section id	
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment	
<b>Description</b>	Sets or removes an instance assignment. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless <code>"-dont_export_assignments"</code> is specified) These two Tcl commands reside in the <code>::quartus::project</code> Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands <code>"execute_flow"</code> and <code>"execute_module"</code> (part of the <code>::quartus::flow</code> Tcl package) automatically call <code>"export_assignments"</code> before they run command-line executables. For entity-specific assignments, use the <code>"-entity"</code> option to force the assignment to specified entity. If the <code>"-entity"</code> option is not specified, the value for the <code>FOCUS_ENTITY_NAME</code> assignment is used. If the <code>FOCUS_ENTITY_NAME</code> value is not found, the revision name is used.		
<b>Example Usage</b>	<pre>## Specify a TSU_REQUIREMENT of 2ns from mypin to any register set_instance_assignment -from "mypad" -to * -name TSU_REQUIREMENT 2ns  ## Remove the TSU_REQUIREMENT from mypin to all registers set_instance_assignment -from "mypad" -to * -name TSU_REQUIREMENT -remove  ## Specify the entity to which the assignment is added, ## use the -entity option ## This is needed if the top-level entity name is other than ## that of the project name ## The following command generates a top_level entity set_instance_assignment -from "mypad" -to * -entity top_level -name TSU_REQUIREMENT 2ns</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: Ignored assignment: <string>. The assignment is no longer supported.

*continued...*

TCL_ERROR	1	ERROR: Assignment is not an instance assignment: <string> -- it is a global assignment. Specify an instance assignment name or use the global assignment commands.
TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
TCL_ERROR	1	ERROR: The assignment <string> is from Design Template and can't be changed/removed.
TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: The -<string> option is not required but was specified with the value: <string>. Delete the option.
TCL_ERROR	1	ERROR: The -<string> option is required but was not specified. Specify the required option.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: An unknown error has occurred.
TCL_ERROR	1	ERROR: Assignment <string> cannot be removed -- it has multiple values. Specify one value to remove or use the <string> command to remove all values for the assignment.
TCL_ERROR	1	ERROR: Missing <<string>> for <string> assignment. Specify the required <string>.

### 3.1.24.51. set\_io\_assignment (::quartus::project)

The following table displays information for the `set_io_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::project</code> on page 257		
<b>Syntax</b>	<code>set_io_assignment [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-fall] [-io_standard &lt;io standard&gt;] -name &lt;name&gt; [-remove] [-rise] [-tag &lt;data&gt;] [&lt;value&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-comment &lt;comment&gt;</code>	Comment	
	<code>-disable</code>	Option to disable assignment	
	<code>-fall</code>	Option applies to falling edge	
	<code>-io_standard &lt;io standard&gt;</code>	Option to specify the io standard	
	<code>-name &lt;name&gt;</code>	Assignment name	
	<code>-remove</code>	Option to remove assignment	
	<code>-rise</code>	Option applies to rising edge	
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment	
	<code>&lt;value&gt;</code>	Assignment value	
<b>Description</b>	Sets or removes an io assignment. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless <code>"-dont_export_assignments"</code> is specified) These two Tcl commands reside in the <code>::quartus::project</code> Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands <code>"execute_flow"</code> and <code>"execute_module"</code> (part of the <code>::quartus::flow</code> Tcl package) automatically call <code>"export_assignments"</code> before they run command-line executables.		
<b>Example Usage</b>	<pre>## Specify LVTTL as the IO Standard for OUTPUT_PIN_LOAD assignment set_io_assignment 30 -name OUTPUT_PIN_LOAD -io_standard LVTTL</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	WARNING: Ignored assignment: <string>. The assignment is no longer supported.
	TCL_ERROR	1	ERROR: Assignment is not a global assignment: <string> -- it is an instance assignment. Specify a global assignment name or use the instance assignment commands.
	TCL_ERROR	1	ERROR: Can't find file(s) associated with assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find section information for assignment. Specify a different assignment name.
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: File name <string> exceeds maximum of <string> characters. Specify a file name with fewer characters.

*continued...*

TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Option -<string> for <string> assignment is illegal. Specify a legal option or remove the option.
TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
TCL_ERROR	1	ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.
TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
TCL_ERROR	1	ERROR: The -<string> option is not required but was specified with the value: <string>. Delete the option.
TCL_ERROR	1	ERROR: The -<string> option is required but was not specified. Specify the required option.
TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run get_all_assignment_names.
TCL_ERROR	1	ERROR: Assignment <string> cannot be removed -- it has multiple values. Specify one value to remove or use the <string> command to remove all values for the assignment.
TCL_ERROR	1	ERROR: Missing <<string>> for <string> assignment. Specify the required <string>.

### 3.1.24.52. set\_location\_assignment (::quartus::project)

The following table displays information for the set\_location\_assignment Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	set_location_assignment [-h   -help] [-long_help] [-comment <comment>] [-disable] [-fall] [-remove] [-rise] [-tag <data>] -to <destination> [ <value> ]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-comment <comment>	Comment
	-disable	Option to disable assignment
	-fall	Option applies to falling edge
	-remove	Option to remove assignment
	-rise	Option applies to rising edge

*continued...*

	-tag <data>	Option to tag data to this assignment	
	-to <destination>	Destination of assignment	
	<value>	Assignment value	
<b>Description</b>	Sets or removes a location assignment. Valid location assignments, and settings for those assignments for your design are determined by the target device, package type, and pin count. To explore possible assignments and settings for your design and device, in the Quartus Prime Assignment Editor, specify Location for Assignment Name, and then click Browse in the Value column. In the Location dialog box, explore assignable device resources in the Element list, and then use the lists that appear (based on the element selected) to determine locations of resources that can be specified as the value for the assignment. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.		
<b>Example Usage</b>	set_location_assignment -to dst LOC		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: The assignment <string> is from Design Template and can't be changed/removed.
	TCL_ERROR	1	ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.
	TCL_ERROR	1	ERROR: Missing destination for assignment. Specify the destination for the assignment.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Missing <> for <string> assignment. Specify the required <string>.

### 3.1.24.53. set\_multicycle\_assignment (::quartus::project)

The following table displays information for the set\_multicycle\_assignment Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	set_multicycle_assignment [-h   -help] [-long_help] [-comment <comment>] [-disable] [-end] [-fall] [-from <from_list>] [-hold] [-remove] [-rise] [-setup] [-start] [-tag <data>] [-to <to_list>] <path_multiplier>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-comment <comment>	Comment
	-disable	Option to disable multicycle assignment
	-end	Option to indicate that destination clock cycles should be considered for path multiplier
	-fall	Option applies to falling edge

*continued...*

	<p>-from &lt;from_list&gt;</p> <p>-hold</p> <p>-remove</p> <p>-rise</p> <p>-setup</p> <p>-start</p> <p>-tag &lt;data&gt;</p> <p>-to &lt;to_list&gt;</p> <p>&lt;path_multiplier&gt;</p>	<p>List of clock names, node names, and/or assignment group names that represent start or source points of multicycle path, for example, {node1 node2 ...}</p> <p>Option to indicate that path multiplier is meant for hold</p> <p>Option to remove multicycle assignment</p> <p>Option applies to rising edge</p> <p>Option to indicate that path multiplier is meant for setup</p> <p>Option to indicate that source clock cycles must be considered for path multiplier</p> <p>Option to tag data to this assignment</p> <p>List of clock names, node names, and/or assignment group names that represent end or destination points of multicycle path, for example, {node1 node2 ...}</p> <p>Multicycle path multiplier</p>
<b>Description</b>	<p>Specifies that the given timing paths have multicycle setup or hold delays with the number of cycles specified by the "-path_multiplier" option. If neither the "-setup" nor "-hold" options are used, the "-setup" option is the default option. If neither the "-start" nor "-end" options are used, the "-end" option is the default option. You must use either the "-from &lt;from_list&gt;" or "-to &lt;to_list&gt;" option. Note that Quartus Prime timing analysis is optimized to use assignment groups for timing constraints instead of a list of nodes. Of the following two methods to make multicycle assignments, method (1) is the optimal method. (1) assignment_group "src_group" -add_member "s1" assignment_group "src_group" -add_member "s2" assignment_group "src_group" -add_member "s3" assignment_group "dst_group" -add_member "d1" assignment_group "dst_group" -add_member "d2" set_multicycle_assignment -from "src_group" -to "dst_group" (2) set_multicycle_assignment -from {s1 s2 s3} -to {d1 d2} For more information about assignment groups, type "assignment_group -h". The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0. The meaning of multicycle hold differs between the Quartus Prime software timing analysis and the Synopsys PrimeTime software timing analysis. Refer to the online Help of each software for more information. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</p>	
<b>Example Usage</b>	<pre>## Multicycle "setup" from reg1 and reg2 to any destination points assignment_group "src_group" -add_member reg1 assignment_group "src_group" -add_member reg2 set_multicycle_assignment 2 -setup -from "src_group"  ## or assignment_group "src_group" -add_member reg1 assignment_group "src_group" -add_member reg2 assignment_group "dst_group" -add_member * set_multicycle_assignment 2 -setup -from "src_group" -to "dst_group"  ## Source multicycle "setup" to reg1 and reg2 from any source points assignment_group "dst_group" -add_member reg1 assignment_group "dst_group" -add_member reg2 set_multicycle_assignment 2 -setup -start -to "dst_group"  ## or assignment_group "dst_group" -add_member reg1 assignment_group "dst_group" -add_member reg2 assignment_group "src_group" -add_member * set_multicycle_assignment 2 -setup -start -from "src_group" -to "dst_group"  ## Source multicycle "hold" from src1 to dst1 and dst2 and ## from src2 to dst1 and dst2 assignment_group "src_group" -add_member src1 assignment_group "src_group" -add_member src2 assignment_group "dst_group" -add_member dst1 assignment_group "dst_group" -add_member dst2 set_multicycle_assignment 2 -hold -from "src_group" -to "dst_group"</pre>	

*continued...*

	<pre>## Source multicycle "hold" from registers clocked by clk1 ## to registers clocked by clk2 ## Timegroups are useful for making assignments to ## more than one node. Timegroups are not necessary ## for making an assignment from only one clock node ## to another clock set_multicycle_assignment 2 -hold -start -from clk1 -to clk2</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't set revision: <string>. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Invalid path multiplier value: <string>. Specify a positive integer value.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
	TCL_ERROR	1	ERROR: Revision does not exist: <string>. Specify a legal revision name using the -<string> option.

### 3.1.24.54. `set_parameter` (::quartus::project)

The following table displays information for the `set_parameter` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 257	
<b>Syntax</b>	<pre>set_parameter [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-entity &lt;entity_name&gt;] [-fall] -name &lt;name&gt; [-remove] [-rise] [-tag &lt;data&gt;] [-to &lt;destination&gt;] [ &lt;value&gt; ]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-comment <comment>	Comment
	-disable	Option to disable parameter
	-entity <entity_name>	Entity to which to add parameter
	-fall	Option applies to falling edge
	-name <name>	Parameter name
	-remove	Option to remove parameter
	-rise	Option applies to rising edge
	-tag <data>	Option to tag data to this assignment
	-to <destination>	Destination of parameter
	<value>	Parameter value

*continued...*

<b>Description</b>	<p>Sets or removes the specified parameter name. The "-name" option is not case sensitive. The "-to" option is case sensitive. The parameters created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables. set_parameter can be used to overwrite the parameters in the top-level entity of the design. A warning will be given if the parameter can not be applied. set_parameter assignments with "-entity" and "-to" are ignored and a critical warning is given if they are used.</p>			
<b>Example Usage</b>	<pre>## Set project-wide, default WIDTH parameter value set_parameter -name WIDTH 8  ## Set entity-specific SIZE parameter value ## to "my_ram" entity set_parameter -entity my_ram -name SIZE 16  ## Specify the same parameter to my_ram ## but inside "top_level" entity set_parameter -entity top_level -to my_ram -name SIZE 16</pre>			
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> </table>	Code Name	Code	String Return
Code Name	Code	String Return		
TCL_OK	0 INFO: Operation successful			
TCL_OK	0 INFO: Removed parameter: <string>			
TCL_ERROR	1 ERROR: Can't find active revision name. Make sure there is an open, active revision name.			
TCL_ERROR	1 ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.			
TCL_ERROR	1 ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.			
TCL_ERROR	1 ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.			
TCL_ERROR	1 ERROR: You must open a project before you can use this command.			
TCL_ERROR	1 ERROR: The -<string> option is not required but was specified with the value: <string>. Delete the option.			
TCL_ERROR	1 ERROR: The -<string> option is required but was not specified. Specify the required option.			
TCL_ERROR	1 ERROR: Parameter does not exist and cannot be removed: <string>. Specify an existing parameter name.			
TCL_ERROR	1 ERROR: An unknown error has occurred.			
TCL_ERROR	1 ERROR: Missing <> for <string> assignment. Specify the required <string>.			

<b>Description</b>	<p>Sets or removes the specified parameter name. The "-name" option is not case sensitive. The "-to" option is case sensitive. The parameters created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables. set_parameter can be used to overwrite the parameters in the top-level entity of the design. A warning will be given if the parameter can not be applied. set_parameter assignments with "-entity" and "-to" are ignored and a critical warning is given if they are used.</p>			
<b>Example Usage</b>	<pre>## Set project-wide, default WIDTH parameter value set_parameter -name WIDTH 8  ## Set entity-specific SIZE parameter value ## to "my_ram" entity set_parameter -entity my_ram -name SIZE 16  ## Specify the same parameter to my_ram ## but inside "top_level" entity set_parameter -entity top_level -to my_ram -name SIZE 16</pre>			
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> </table>	Code Name	Code	String Return
Code Name	Code	String Return		
TCL_OK	0 INFO: Operation successful			
TCL_OK	0 INFO: Removed parameter: <string>			
TCL_ERROR	1 ERROR: Can't find active revision name. Make sure there is an open, active revision name.			
TCL_ERROR	1 ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.			
TCL_ERROR	1 ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.			
TCL_ERROR	1 ERROR: Value <string> for <string> assignment is illegal. Specify a legal value.			
TCL_ERROR	1 ERROR: You must open a project before you can use this command.			
TCL_ERROR	1 ERROR: The -<string> option is not required but was specified with the value: <string>. Delete the option.			
TCL_ERROR	1 ERROR: The -<string> option is required but was not specified. Specify the required option.			
TCL_ERROR	1 ERROR: Parameter does not exist and cannot be removed: <string>. Specify an existing parameter name.			
TCL_ERROR	1 ERROR: An unknown error has occurred.			
TCL_ERROR	1 ERROR: Missing <> for <string> assignment. Specify the required <string>.			

### 3.1.24.55. set\_power\_file\_assignment (::quartus::project)

The following table displays information for the `set_power_file_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::project</code> on page 257				
<b>Syntax</b>	<code>set_power_file_assignment [-h   -help] [-long_help] [-remove] [-saf_file &lt;saf_file&gt;] [-section_id &lt;section_id&gt;] [-to &lt;to&gt;] [-vcd_end_time &lt;vcd_end_time&gt;] [-vcd_file &lt;vcd_file&gt;] [-vcd_start_time &lt;vcd_start_time&gt;]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-remove</code>	Option to remove assignment			
	<code>-saf_file &lt;saf_file&gt;</code>	SAF file name			
	<code>-section_id &lt;section_id&gt;</code>	Section id			
	<code>-to &lt;to&gt;</code>	Entity to which to apply power input file			
	<code>-vcd_end_time &lt;vcd_end_time&gt;</code>	End time for VCD file parsing			
	<code>-vcd_file &lt;vcd_file&gt;</code>	VCD file name			
	<code>-vcd_start_time &lt;vcd_start_time&gt;</code>	Start time for VCD file parsing			
<b>Description</b>	<p>Sets or removes a power input file assignment. Power input file assignments are specified using multiple global assignments, and a single instance assignment as illustrated in the following example:</p> <pre>set_global_assignment -name POWER_INPUT_FILE_NAME "test.vcd" -section_id test.vcd set_global_assignment -name POWER_INPUT_FILE_TYPE VCD -section_id test.vcd set_global_assignment -name POWER_VCD_FILE_START_TIME "10 ns" -section_id test.vcd set_global_assignment -name POWER_VCD_FILE_END_TIME "1000 ns" -section_id test.vcd set_instance_assignment -name POWER_READ_INPUT_FILE test.vcd -to test_design</pre> <p>The power input file assignment serves as a wrapper for all of the above assignments. If the "-remove" setting is not set, the <code>set_power_file_assignment</code> will also make the following assignment to enable the use of input files:</p> <pre>set_global_assignment -name POWER_USE_INPUT_FILES ON</pre> <p>If you do not specify a "-section_id", a new section identifier is created for the input file assignment. If a "-section_id" is specified and it does not already exist, it is used as the new section identifier. If a "-section_id" is specified and it does exist, the existing input file assignments are removed and a new input file assignment is created using the given parameters and section identifier. If an entity name given by "-to" is not specified, the input file assignment applies to the top level design entity. If the "-remove" setting is used, the input file assignment given by the "-section_id", "-vcd_file", or "-saf_file" is removed from the project. Assignments created or modified by using this Tcl command are saved to the Quartus Prime Settings File (.qsf).</p>				
<b>Example Usage</b>	<pre>## Specify an input SAF file applied to the top level entity ## A default section will be created set_power_file_assignment -saf_file test.saf  ## Specify an input VCD file applied to design_top counter1 ## Use the given section_id to create a new section set_power_file_assignment -vcd_file test.vcd -to design_top counter1 -section_id test.vcd  ## Update the previous input VCD file assignment to specify a ## start and end time set_power_file_assignment -vcd_file test.vcd -to design_top counter1 -vcd_start_time 10ns - vcd_end_time 100ns -section_id test.vcd  ## Remove the input SAF file assignment using the file name set_power_file_assignment -saf_file test.saf -remove  ## Remove the input VCD file assignment using the section identifier set_power_file_assignment -section_id test.vcd -remove</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

*continued...*

TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
TCL_ERROR	1	ERROR: Compiler database does not exist for revision name: <string>. At the minimum, run Analysis and Synthesis (quartus_map) with the specified revision name before using this Tcl command.
TCL_ERROR	1	ERROR: Exactly one of the following file name options must be specified: -<string> or -<string>.
TCL_ERROR	1	ERROR: If -<string> is set, exactly one of the following options must be specified: -<string>, -<string> or -<string>. All other options must not be set.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: -<string> and -<string> cannot be used with -<string> option.

### 3.1.24.56. set\_project\_settings (::quartus::project)

The following table displays information for the `set_project_settings` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::project</code> on page 0		
<b>Syntax</b>	<code>set_project_settings [-h   -help] [-long_help] [-cmp &lt;revision_name&gt;] [-sim &lt;revision_name&gt;] [-swb &lt;revision_name&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cmp <revision_name>	Revision name	
	-sim <revision_name>	Revision name	
	-swb <revision_name>	Revision name	
<b>Description</b>	Sets the current revision to the value specified by the "-cmp", "-sim", or "-swb" option.		
<b>Example Usage</b>	<pre>## Sets the revision to auto_max.qsf set_project_settings -cmp auto_max  ## Sets the revision to auto_max.qsf set_project_settings -sim auto_max  ## Sets the revision to auto_max.qsf set_project_settings -swb auto_max</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't create revision: <string>. Specify a legal revision name using the -<string> option.
	TCL_ERROR	1	ERROR: Can't run Tcl command while a process is in progress: <string>. To run the command, stop the compilation or simulation; or wait for the compilation or simulation to complete.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.24.57. set\_timing\_cut\_assignment (::quartus::project)

The following table displays information for the `set_timing_cut_assignment` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::project on page 0	
<b>Syntax</b>	<code>set_timing_cut_assignment [-h   -help] [-long_help] [-comment &lt;comment&gt;] [-disable] [-fall] [-from &lt;from_pin_list&gt;] [-remove] [-rise] [-tag &lt;data&gt;] [-to &lt;to_pin_list&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-comment &lt;comment&gt;</code>	Comment
	<code>-disable</code>	Option to disable assignment
	<code>-fall</code>	Option applies to falling edge
	<code>-from &lt;from_pin_list&gt;</code>	List of start or source node names and/or assignment group names for timing path, for example, {node1 node2 ...}
	<code>-remove</code>	Option to remove timing cut assignment
	<code>-rise</code>	Option applies to rising edge
	<code>-tag &lt;data&gt;</code>	Option to tag data to this assignment
	<code>-to &lt;to_pin_list&gt;</code>	List of end node names and/or assignment group names for timing path, for example, {node1 node2 ...}
<b>Description</b>	Specifies that the timing paths that start from the designated <code>&lt;from_pin_list&gt;</code> and end in the designated <code>&lt;to_pin_list&gt;</code> are false paths. Nodes for the <code>&lt;from_pin_list&gt;</code> can be input pins, internal nodes, clock pins, or assignment groups. Nodes for the <code>&lt;to_pin_list&gt;</code> can be output pins, internal nodes, clock pins, or assignment groups. You must use either the " <code>-from &lt;from_pin_list&gt;</code> " or the " <code>-to &lt;to_pin_list&gt;</code> " option. Note that Quartus Prime timing analysis is optimized to use assignment groups for timing constraints instead of a list of nodes. Of the following two methods to make timing cut assignments, method (1) is the optimal method. (1) <code>assignment_group "src_group" -add_member "s1" assignment_group "src_group" -add_member "s2" assignment_group "src_group" -add_member "s3" assignment_group "dst_group" -add_member "d1" assignment_group "dst_group" -add_member "d2" set_timing_cut_assignment -from "src_group" -to "dst_group"</code> (2) <code>set_timing_cut_assignment -from {s1 s2 s3} -to {d1 d2}</code> For more information about assignment groups, type " <code>assignment_group -h</code> ". The " <code>assignment_group</code> " command replaces the deprecated " <code>timegroup</code> " command in ::quartus::project, version 5.0. Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) <code>export_assignments</code> 2) <code>project_close</code> (unless " <code>-dont_export_assignments</code> " is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands " <code>execute_flow</code> " and " <code>execute_module</code> " (part of the ::quartus::flow Tcl package) automatically call " <code>export_assignments</code> " before they run command-line executables.	
<b>Example Usage</b>	<pre>## Set timing cut from any source points to dst1 and dst2 assignment_group "dst_group" -add_member dst1 assignment_group "dst_group" -add_member dst2 set_timing_cut_assignment -to "dst_group"  ## or assignment_group "src_group" -add_member * assignment_group "dst_group" -add_member dst1 assignment_group "dst_group" -add_member dst2 set_timing_cut_assignment -from "src_group" -to "dst_group"  ## Set timing cut from src1 and src2 to any end points assignment_group "src_group" -add_member src1 assignment_group "src_group" -add_member src2 set_timing_cut_assignment -from "src_group"  ## or assignment_group "src_group" -add_member src1</pre>	

*continued...*

	<pre>assignment_group "src_group" -add_member src2 assignment_group "dst_group" -add_member * set_timing_cut_assignment -from "src_group" -to "dst_group"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't set revision: <string>. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
	TCL_ERROR	1	ERROR: Revision does not exist: <string>. Specify a legal revision name using the -<string> option.

### 3.1.24.58. `set_user_option` (::quartus::project)

The following table displays information for the `set_user_option` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257				
<b>Syntax</b>	<code>set_user_option [-h   -help] [-long_help] -name &lt;name&gt; [ &lt;value&gt; ]</code>				
<b>Arguments</b>	-h   -help	Short help			
	-long_help	Long help with examples and possible return values			
	-name <name>	User option name			
	<value>	User option value			
<b>Description</b>	Sets the user option value for the name specified by the "-name" option. The user option is written to the quartus2.ini file. To get a list of all available user option names, use the "get_all_user_option_names" command.				
<b>Example Usage</b>	<pre>## Set TALKBACK_ENABLED to "on" set_user_option -name TALKBACK_ENABLED on</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Illegal user option name: <string>. Specify a legal user option name. To get a list of legal names, use the get_all_user_option_names command.		
	TCL_ERROR	1	ERROR: Illegal user option value: <string>. Specify a legal user option value.		

### 3.1.24.59. test\_assignment\_trait (::quartus::project)

The following table displays information for the `test_assignment_trait` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 257		
<b>Syntax</b>	<code>test_assignment_trait [-h   -help] [-long_help] -name &lt;name&gt; -trait &lt;trait_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	Assignment name	
	-trait <trait_name>	Trait name	
<b>Description</b>	Checks whether the assignment name has the specified trait. Returns 1, if the assignment name has the trait; returns 0, otherwise.		
<b>Example Usage</b>	<pre>## Test if the assignment name is case-sensitive if {[test_assignment_trait -name VHDL_FILE -trait CASE_SENSITIVE]} {     puts "VHDL_FILE assignment is case-sensitive." } else {     puts "VHDL_FILE assignment is not case-sensitive." }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Assignment <string> is not supported in this edition of the Quartus Prime software.
	TCL_ERROR	1	ERROR: Value <string> for the -<string> option is illegal. Specify a legal value.
	TCL_ERROR	1	ERROR: Options cannot be specified together: -<string>, -<string> and -<string>. Specify only one or two of the three options.
	TCL_ERROR	1	ERROR: Ignored assignment: <string>. The assignment is no longer supported.
	TCL_ERROR	1	ERROR: Illegal assignment name: <string>. Specify a legal assignment name. To view the list of legal assignment names, run <code>get_all_assignment_names</code> .
	TCL_ERROR	1	ERROR: Illegal trait: <string>. Specify a legal trait name.

### 3.1.24.60. timegroup (::quartus::project)

The following table displays information for the `timegroup` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::project</a> on page 0		
<b>Syntax</b>	<code>timegroup [-h   -help] [-long_help] [-add_exception &lt;name&gt;] [-add_member &lt;name&gt;] [-comment &lt;comment&gt;] [-disable] [-get_exceptions] [-get_members] [-overwrite] [-remove] [-remove_exception &lt;name&gt;] [-remove_member &lt;name&gt;] &lt;group_name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	

*continued...*

	<ul style="list-style-type: none"> <li>-add_exception &lt;name&gt; Tcl list of exception names to add</li> <li>-add_member &lt;name&gt; Tcl list of member names to add</li> <li>-comment &lt;comment&gt; Comment</li> <li>-disable Option to disable assignment</li> <li>-get_exceptions Option to get collection of timegroup exceptions</li> <li>-get_members Option to get collection of timegroup members</li> <li>-overwrite Option to overwrite existing timegroup with the same group name</li> <li>-remove Option to remove timegroup</li> <li>-remove_exception &lt;name&gt; Tcl list of exception names to remove</li> <li>-remove_member &lt;name&gt; Tcl list of member names to remove</li> <li>&lt;group_name&gt; Group name</li> </ul>
<b>Description</b>	<p>Adds, removes, gets members of, or gets exceptions to a timegroup. A timegroup is a custom group of registers and pins. You can use the "-add_member" option to specify register or pin names you want to include in the timegroup. You can use the "-add_exception" option to specify names you want to exclude from the timegroup. You can specify the names using wildcards, that is, using "?" or "*". For example, to add all registers and pins that start with a "b" except those that start with "b c " to a particular timegroup named "group_b", type: timegroup "group_b" -add_member "b*" -add_exception "b c *" To remove members or exceptions from a timegroup, use the "-remove_member" or "-remove_exception" options respectively. The "-get_members" option returns a collection of members in the timegroup. The "-get_exceptions" option returns a collection of exceptions to the timegroup. To access each element of the collection, use the Tcl command "foreach_in_collection". To see example usage, type "timegroup -long_help" or "foreach_in_collection -long_help". Specifying registers and pins in terms of a timegroup allows you to set timing constraints easily. For example, to make a multicycle assignment from nodes "a1" and "a2" to nodes "b1", "b2", and "b3", type the following: timegroup "group_a" -add_member [list "a1" "a2"] timegroup "group_b" -add_member [list "b1" "b2" "b3"] set_multicycle_assignment -from "group_a" -to "group_b" 2 This command sets a multicycle assignment from every member of "group_a" to every member of "group_b". Quartus Prime timing analysis is optimized to use timegroups in handling timing constraints. To disable timegroup assignments for the entire group, use the "-disable" option, for example: timegroup "group_a" -disable To disable a particular timegroup assignment, use the "-disable" option with the "-add_member" or "-add_exception" options, for example: timegroup "group_a" -add_member "m1" -disable timegroup "group_a" -add_exception "e1" -disable Assignments created or modified by using this Tcl command are not saved to the Quartus Prime Settings File (.qsf) unless you explicitly call one of the following two Tcl commands: 1) export_assignments 2) project_close (unless "-dont_export_assignments" is specified) These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus Prime command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.</p>
<b>Example Usage</b>	<pre># Make timing cut assignment from nodes starting # with "r" except those starting with "r s" # and except those starting with "r t " # to nodes "t1", "t2", and "t3" timegroup "tg1" -add_member "r*" -add_exception "r s *" timegroup "tg1" -add_exception "r t *"  timegroup "tg2" -add_member [list "t1" "t2" "t3"] set_timing_cut_assignment -from "group_a" -to "group_b" 2  # Remove the "t1" from a particular timegroup named "tg2" timegroup "tg2" -remove_member "t1"  # Display the members of a particular timegroup named "tg1" foreach_in_collection member [timegroup "tg1" -get_members] {     # Print the name of the member     puts \$member }  # Display the exceptions to a particular timegroup named "tg1"</pre>

*continued...*

	<pre>foreach_in_collection exception [timegroup "tg1" -get_exceptions] {     # Print the name of the exception     puts \$exception }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find active revision name. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: Entity does not exist or uses illegal name characters: <string>. Specify a legal entity name.
	TCL_ERROR	1	ERROR: Can't set revision: <string>. Make sure there is an open, active revision name.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Found two options: -<string> and -<string>. Choose one of the options.
	TCL_ERROR	1	ERROR: Revision does not exist: <string>. Specify a legal revision name using the -<string> option.

### 3.1.25. ::quartus::qshm

The following table displays information for the **::quartus::qshm** Tcl package:

<b>Tcl Package and Version</b>	::quartus::qshm 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is loaded by default in the following executables:  qpro_sh quartus_sh
<b>Tcl Commands</b>	<a href="#">qshm_connect_to_quartus</a> (::quartus::qshm) on page 323 <a href="#">qshm_disconnect_from_quartus</a> (::quartus::qshm) on page 324 <a href="#">qshm_dispose_client</a> (::quartus::qshm) on page 324 <a href="#">qshm_get_hub_key_prefix</a> (::quartus::qshm) on page 325 <a href="#">qshm_get_parent_hub_key</a> (::quartus::qshm) on page 325 <a href="#">qshm_obtain_client</a> (::quartus::qshm) on page 325 <a href="#">qshm_send_request</a> (::quartus::qshm) on page 326 <a href="#">qshm_send_server_state_query</a> (::quartus::qshm) on page 326 <a href="#">qshm_set_context</a> (::quartus::qshm) on page 327

#### 3.1.25.1. qshm\_connect\_to\_quartus (::quartus::qshm)

The following table displays information for the `qshm_connect_to_quartus` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::qshm</a> on page 323	
<b>Syntax</b>	<code>qshm_connect_to_quartus [-h   -help] [-long_help] -hub_key_prefix &lt;hub_key_prefix&gt; -name &lt;name&gt; [-project &lt;project&gt;] [-revision &lt;revision&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-hub_key_prefix <hub_key_prefix>	hub key prefix

*continued...*

	-name <name>	client name	
	-project <project>	Quartus Prime project	
	-revision <revision>	Quartus Prime project revision	
<b>Description</b>	Connects a client with the provided name to a Quartus Prime hub and returns the key of the dedicated server created by the hub for that client.		
<b>Example Usage</b>	<pre>## Connect a client with name client1 to a Quarts Prime hub with hub key prefix hub1 qshm_connect_to_quartus -name client1 -hub_key_prefix hub1 -project &lt;project&gt; -revision &lt;revision&gt;</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.2. qshm\_disconnect\_from\_quartus (::quartus::qshm)

The following table displays information for the qshm\_disconnect\_from\_quartus Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	qshm_disconnect_from_quartus [-h   -help] [-long_help] -name <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	client name	
<b>Description</b>	Disconnect a client with the specified name from the Quarts Prime it is connected to.		
<b>Example Usage</b>	<pre>## Disconnect a client with name client1 from the Quarts Prime qshm_disconnect_from_quartus -name client1</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.3. qshm\_dispose\_client (::quartus::qshm)

The following table displays information for the qshm\_dispose\_client Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	qshm_dispose_client [-h   -help] [-long_help] [-name <name> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	Client name	
<b>Description</b>	Dispose and reclaim resources by the client with the specified name.		
<b>Example Usage</b>	<pre>## Dispose of a client with name client1 qshm_dispose_client -name client1</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.4. qshm\_get\_hub\_key\_prefix (::quartus::qshm)

The following table displays information for the `qshm_get_hub_key_prefix` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	<code>qshm_get_hub_key_prefix [-h   -help] [-long_help] [-edition &lt;edition&gt;] [-qroot_dir &lt;qroot_dir&gt;] [-version &lt;version&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-edition <edition>	Quartus Prime edition	
	-qroot_dir <qroot_dir>	Quartus Prime root directory	
	-version <version>	Quartus Prime version	
<b>Description</b>	Returns the key prefix for a Quartus Prime hub based on the specified configuration arguments.		
<b>Example Usage</b>	<pre>## Create an SHM-based server type socket with the port name "trip" qshm_get_hub_key_prefix -qroot_dir /a/b/c -version 16.1 -edition "Quartus Prime Pro"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.5. qshm\_get\_parent\_hub\_key (::quartus::qshm)

The following table displays information for the `qshm_get_parent_hub_key` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	<code>qshm_get_parent_hub_key [-h   -help] [-long_help] -name &lt;name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	client name	
<b>Description</b>	Obtains the complete SHM key of the parent hub of the server that is connected to the client specified by the name argument		
<b>Example Usage</b>			
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.6. qshm\_obtain\_client (::quartus::qshm)

The following table displays information for the `qshm_obtain_client` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	<code>qshm_obtain_client [-h   -help] [-long_help] [-name &lt;name&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
<i>continued...</i>			

	-long_help	Long help with examples and possible return values			
	-name <name>	Client name			
<b>Description</b>	Creates and returns a client for the specified client name.				
<b>Example Usage</b>	<pre>## Create an SHM-based client type socket with the name "client1" qshm_obtain_client -name client1</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.25.7. qshm\_send\_request (::quartus::qshm)

The following table displays information for the qshm\_send\_request Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	qshm_send_request [-h   -help] [-long_help] -cmd <cmd> -name <name>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-cmd <cmd>	command string	
	-name <name>	client name	
<b>Description</b>	Sends the a request to the client with the specified name and waits to receive a response from the dedicated server the client is connected to.		
<b>Example Usage</b>	<pre>## Send a Tcl command to the client with the specified name. The context should have been set to include the request_type=tcl key value pair. qshm_send_request -name client1 -cmd "set x 5"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.8. qshm\_send\_server\_state\_query (::quartus::qshm)

The following table displays information for the qshm\_send\_server\_state\_query Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	qshm_send_server_state_query [-h   -help] [-long_help] -name <name> -state_key <state_key>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <name>	client name	
	-state_key <state_key>	state key string	
<b>Description</b>	Sends the a server state request to the client with the specified name and waits to receive a response from the state socket of the dedicated server the client is connected to.		

*continued...*

<b>Example Usage</b>	<pre>## Send a server state command using the client with specified name. qshm_send_server_state_query -name client1 -state_key device_family</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.25.9. qshm\_set\_context (::quartus::qshm)

The following table displays information for the qshm\_set\_context Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::qshm on page 323		
<b>Syntax</b>	<pre>qshm_set_context [-h   -help] [-long_help] -key &lt;key&gt; -name &lt;name&gt; -value &lt;value&gt;</pre>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-key <key>	A context key	
	-name <name>	client name	
	-value <value>	A context value	
<b>Description</b>	Sets the a context key value pair for the client with the specified name.		
<b>Example Usage</b>	<pre>## Set a key value pair request_type=tcl for the client with the specified name. qshm_set_context -name client1 -key request_type -value tcl</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.26. ::quartus::report

The following table displays information for the ::quartus::report Tcl package:

<b>Tcl Package and Version</b>	::quartus::report 2.1
<b>Description</b>	This package contains a set of Tcl functions for accessing and updating information in the report database.
<b>Availability</b>	<p>This package is loaded by default in the following executables:</p> <pre>gacv quartus_fit quartus_ipgenerate quartus_sim quartus_stp</pre> <p>This package is available for loading in the following executables:</p> <pre>gpro gpro_sh quartus quartus_cdb quartus_drc quartus_eda quartus_map quartus_sh quartus_si quartus_sta quartus_syn</pre>
<b>Tcl Commands</b>	<a href="#">add_row_to_table (::quartus::report)</a> on page 328 <a href="#">create_report_panel (::quartus::report)</a> on page 329 <a href="#">delete_report_panel (::quartus::report)</a> on page 330

*continued...*

```

get_fitter_resource_usage (::quartus::report) on page 332
get_number_of_columns (::quartus::report) on page 333
get_number_of_rows (::quartus::report) on page 334
get_report_panel_column_index (::quartus::report) on page 335
get_report_panel_data (::quartus::report) on page 336
get_report_panel_id (::quartus::report) on page 337
get_report_panel_names (::quartus::report) on page 338
get_report_panel_row (::quartus::report) on page 339
get_report_panel_row_index (::quartus::report) on page 340
load_report (::quartus::report) on page 341
read_xml_report (::quartus::report) on page 342
refresh_report_window (::quartus::report) on page 343
save_report_database (::quartus::report) on page 343
unload_report (::quartus::report) on page 344
write_report_panel (::quartus::report) on page 344
write_xml_report (::quartus::report) on page 346

```

### **3.1.26.1. add\_row\_to\_table (::quartus::report)**

The following table displays information for the `add_row_to_table` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>add_row_to_table [-h   -help] [-long_help] [-id &lt;table_id&gt;] [-name &lt;table_name&gt;] &lt;row&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-id &lt;table_id&gt;</code>	Id of table to update	
	<code>-name &lt;table_name&gt;</code>	Name of table to update	
	<code>&lt;row&gt;</code>	Tcl list of strings to add to table	
<b>Description</b>	Adds the list of strings to the table as a row. Using the panel id provides faster data access than using the panel name. Panel ids that you have cached may become outdated or invalid if the report gets unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of content portion of the UI Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the UI Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Set panel name and id set panel {Fitter  Fitter Settings} set id [get_report_panel_id \$panel]  if {\$id != -1} {     # If panel exists, add a row to it     add_row_to_table -id \$id {{New Field} Yes No}     # Save the changes to the report database     save_report_database } else {     # Otherwise print an error message     puts "Error: Table \$panel does not exist." }  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Number of elements conflict. The number of elements in the added row must be <string>.

*continued...*

TCL_ERROR	1	ERROR: Illegal color: <string>. Specify a color that is currently supported by add_row_to_table.
TCL_ERROR	1	ERROR: You specified <string> colors for the -<string> option. However, you must specify <string> colors to match the number of elements specified for the <><string>> argument.
TCL_ERROR	1	ERROR: Value specified for -<string> option is not a valid Tcl list: <string>. Specify a valid Tcl list.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.
TCL_ERROR	1	ERROR: Can't access report panel. Use this command only for report panels with rows or columns.
TCL_ERROR	1	ERROR: Value specified for <row> is not a valid Tcl list: <string>. Specify <row> as a valid Tcl list.
TCL_ERROR	1	ERROR: add_row_to_table command is not allowed for this report panel. You cannot add rows to this report panel.

### 3.1.26.2. create\_report\_panel (::quartus::report)

The following table displays information for the `create_report_panel` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327	
<b>Syntax</b>	<code>create_report_panel [-h   -help] [-long_help] [-folder] [-table] &lt;panel_name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-folder</code>	Option to create folder
	<code>-table</code>	Option to create table
	<code>&lt;panel_name&gt;</code>	Name of the panel to create
<b>Description</b>	Creates a new report panel with the specified name. If -table option is specified, a table is created. If -folder option is specified, a folder is created. The name must be the full path to the new report panel, such as "Fitter  My Table". If the specified panel is created successfully, the corresponding panel id is returned. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".	
<b>Example Usage</b>	<pre>load_package report project_open chiptrip load_report  # Set folder name and id set folder "My Folder" set folder_id [get_report_panel_id \$folder]  # Check if specified folder exists. If not, create it. if {\$folder_id == -1} {     set folder_id [create_report_panel -folder \$folder] }</pre>	

*continued...*

```

# Set table name and id
set table   "$folder|My Table"
set table_id [get_report_panel_id $table]

# Check if specified table exists. If so, delete it.
if {$table_id != -1} {
    delete_report_panel -id $table_id
}

# Create the specified table and get its id
set table_id [create_report_panel -table $table]

# Add Timing Analyzer Summary to the table
add_row_to_table -id $table_id {{Name} {Value}}
add_row_to_table -id $table_id {{Number of Registers} {100}}

# Save the changes to the report database
save_report_database

unload_report
project_close
  
```

Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Report folder <string> already exists -- specify a different report folder name.
TCL_ERROR	1		ERROR: Can't find folder: <string>. Specify an existing report folder name.
TCL_ERROR	1		ERROR: Illegal color: <string>. Specify a color that is currently supported by add_row_to_table.
TCL_ERROR	1		ERROR: The input report object is not a table to be the master table of master details object
TCL_ERROR	1		ERROR: The parent folder is not master details type <string>.
TCL_ERROR	1		ERROR: The master details object already has the master table
TCL_ERROR	1		ERROR: Options -table, -tcl_table and -folder are mutually exclusive -- specify only one of the options.
TCL_ERROR	1		ERROR: You must open a project before you can use this command.
TCL_ERROR	1		ERROR: Report object is not a folder: <string>. Specify an existing report folder name.
TCL_ERROR	1		ERROR: Specify one of the options: -<string> or -<string>.
TCL_ERROR	1		ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.
TCL_ERROR	1		ERROR: Report panel already exists: <string>. Specify a different report panel name.

### 3.1.26.3. delete\_report\_panel (::quartus::report)

The following table displays information for the delete\_report\_panel Tcl command:

Tcl Package and Version	Belongs to ::quartus::report on page 327
Syntax	delete_report_panel [-h   -help] [-long_help] [-id <panel_id>] [-name <panel_name>]
<i>continued...</i>	

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-id &lt;panel_id&gt;</code>	Id of panel to delete	
	<code>-name &lt;panel_name&gt;</code>	Name of panel to delete	
<b>Description</b>	Deletes the report panel with the specified id or name. The panel can either be a report table or report folder. Using the panel id provides faster data access than using the panel name. Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Set table name and id set table "Fitter  My Table" set table_id [get_report_panel_id \$table]  # Delete the table if the it already exists if {\$table_id != -1} {     delete_report_panel -id \$table_id }  # Re-create the table create_report_panel -table \$table add_row_to_table -name \$table {{Name} {Value}} add_row_to_table -name \$table {{Number of Registers} {100}}  # This time, use table name instead of table id to delete it. delete_report_panel -name \$table  # Now, delete a folder set folder "My Folder" set folder_id [get_report_panel_id \$folder]  # Delete it if the specified folder already exists if {\$folder_id != -1} {     delete_report_panel -id \$folder_id }  # Save the changes to the report database save_report_database  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Found conflicting panel options. Specify only one of the options: -name or -id.
	TCL_ERROR	1	ERROR: The delete_report_panel command is not allowed for this report panel. You cannot delete a report panel folder.
	TCL_ERROR	1	ERROR: Illegal panel id: <string>. Specify a legal panel id.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Specify one of the options: -name or -id.
	TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
	TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.

### 3.1.26.4. `get_fitter_resource_usage` (::quartus::report)

The following table displays information for the `get_fitter_resource_usage` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>get_fitter_resource_usage [-h   -help] [-long_help] [-alm] [-alut] [-available] [-io_pin] [-lab] [-le] [-mem_bit] [-percentage] [-reg] [-used] [-utilization]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-alm</code>	Get total adaptive logic modules	
	<code>-alut</code>	Get total Combinational ALUTs	
	<code>-available</code>	Get available resource summary	
	<code>-io_pin</code>	Get total I/O pins	
	<code>-lab</code>	Get total logic array blocks	
	<code>-le</code>	Get total logic elements	
	<code>-mem_bit</code>	Get total memory bits	
	<code>-percentage</code>	Get used resource summary in percentage	
	<code>-reg</code>	Get total registers	
	<code>-used</code>	Get used resource summary	
<b>Description</b>	Gets the Fitter resource usage results. You must use one of the following options: "-alut", "-reg", "-le", "-alm", "-lab", "-io_pin", "-mem_bit" or "-resource". If the above option is not "-resource", you may also optionally use one of the following options: "-used", "-available" or "-percentage". Option "-resource" takes resource name as parameter, which supports wildcards.		
<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Shortcut of get_fitter_resource_usage command set cmd      get_fitter_resource_usage  # Get total registers, logic elements, and DSP block 9-bit elements. set registers [\$cmd -reg] set les        [\$cmd -le -used] set io_pin    [\$cmd -io_pin -available] set mem_bit   [\$cmd -mem_bit -percentage] set dsp       [\$cmd -resource "DSP block 9*"] puts "Registers usage: \$registers" puts "Total used logic elements: \$les" puts "Total available I/O pins: \$io_pin" puts "Total used memory bits in percentage: \${mem_bit}%"  puts "DSP block 9-bit elements: \$dsp"  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Command requires one of the following options: -alut, -reg, -le, -alm, -lab, -io_pin, mem_bit or -resource. Specify one of the options.

*continued...*

TCL_ERROR	1	ERROR: Options -used, -available and -percentage can't be used together. Specify either one or none of the options.
TCL_ERROR	1	ERROR: Option -resource was used with option -used, -available or -percentage. Use option -resource only.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Make sure the project was compiled by quartus_fit and the panel was not deleted.

### 3.1.26.5. get\_number\_of\_columns (::quartus::report)

The following table displays information for the get\_number\_of\_columns Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327		
<b>Syntax</b>	get_number_of_columns [-h   -help] [-long_help] [-id <table_id>] [-name <table_name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <table_id>	Id of panel from which to get data	
	-name <table_name>	Name of panel from which to get data	
<b>Description</b>	Returns the number of columns for the specified panel. Using the panel id provides faster data access than using the panel name. Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre>## Loop through a panel row and print out all its element information load_package report project_open chiptrip load_report  # Set panel name and id set panel {*Input Pins} set id [get_report_panel_id \$panel]     if {\$id == -1} {         error( "Panel not found")     }  # Get the number of columns set col_cnt [get_number_of_columns -id \$id]  # Set row name and get row index set rname {*clk1*} set rindex [get_report_panel_row_index -id \$id \$rname] set rname [get_report_panel_data -row \$rindex -col 0 -id \$id]  puts "\[Input Pins - \$rname\]" for {set i 1} {\$i &lt; \$col_cnt} {incr i} {     set result "[get_report_panel_data -row 0 -col \$i -id \$id]: "     append result [get_report_panel_data -row \$rindex -col \$i -id \$id]     puts \$result }  unload_report project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
TCL_ERROR	1	ERROR: Can't find the top-level panel. Make sure the loaded report is not empty. You can run a successful compilation to rebuild the report.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.
TCL_ERROR	1	ERROR: Can't access report panel. Use this command only for report panels with rows or columns.

### 3.1.26.6. get\_number\_of\_rows (::quartus::report)

The following table displays information for the get\_number\_of\_rows Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327		
<b>Syntax</b>	get_number_of_rows [-h   -help] [-long_help] [-id <table_id>] [-name <table_name>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-id <table_id>	Id of panel from which to get data	
	-name <table_name>	Name of panel from which to get data	
<b>Description</b>	Returns the number of rows for the specified panel. Using the panel id provides faster data access than using the panel name. Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre>## Loop through a panel and print out all its row information load_package report project_open chiptrip load_report  # Set panel name and id set panel {*Fitter Settings} set id [get_report_panel_id \$panel]  # Get the number of rows set row_cnt [get_number_of_rows -id \$id]  for {set i 0} {\$i &lt; \$row_cnt} {incr i} {     puts [get_report_panel_row -row \$i -id \$id] }  unload_report project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
TCL_ERROR	1	ERROR: Can't find the top-level panel. Make sure the loaded report is not empty. You can run a successful compilation to rebuild the report.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.
TCL_ERROR	1	ERROR: Can't access report panel. Use this command only for report panels with rows or columns.

### 3.1.26.7. get\_report\_panel\_column\_index (::quartus::report)

The following table displays information for the get\_report\_panel\_column\_index Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327	
<b>Syntax</b>	get_report_panel_column_index [-h   -help] [-long_help] [-id <table_id>] [-name <table_name>] <col_name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-id <table_id>	Id of panel from which to get column index
	-name <table_name>	Name of panel from which to get column index
	<col_name>	Column name
<b>Description</b>	Gets the column index of the specified panel column name. Column name refers to the specified name in the header row. The column index is a non-negative integer for an existing column. The column index is -1 if the column name is not found in the panel. Using the column index and panel id provides faster data access than using column name and panel name. Column indices and panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Column and panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".	
<b>Example Usage</b>	<pre>load_package report project_open chiptrip load_report  # Set panel name and id set panel {*Input Pins} set id [get_report_panel_id \$panel]  # Set column name and index set cname {Pin #} set cindex [get_report_panel_column_index -id \$id \$cname]  # Get data out of the specified panel set clock [get_report_panel_data -id \$id -row_name clk1 -col \$cindex] set enable [get_report_panel_data -id \$id -row_name enable -col \$cindex]  # Output results puts "\$cname of clock: \$clock" puts "\$cname of enable: \$enable"</pre>	

*continued...*

	unload_report project_close		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.26.8. `get_report_panel_data` (::quartus::report)

The following table displays information for the `get_report_panel_data` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327	
<b>Syntax</b>	<code>get_report_panel_data [-h   -help] [-long_help] [-col &lt;column&gt;] [-col_name &lt;column_name&gt;] [-id &lt;table_id&gt;] [-name &lt;table_name&gt;] [-row &lt;row&gt;] [-row_name &lt;row_name&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-col &lt;column&gt;</code>	column (or X) coordinate
	<code>-col_name &lt;column_name&gt;</code>	column (or X) name
	<code>-id &lt;table_id&gt;</code>	id of panel from which to get data
	<code>-name &lt;table_name&gt;</code>	Name of panel from which to get data
	<code>-row &lt;row&gt;</code>	row (or Y) coordinate
	<code>-row_name &lt;row_name&gt;</code>	row (or Y) name
<b>Description</b>	Returns non-empty data for the specified row and column of the specified panel. If the data is empty or if the row, column, or panel do not exist, an error will be generated. To properly handle the error, make sure to catch the result as in the following example: if [catch {set data [get_report_panel_data ...]} result] { puts "No data found" } else { puts "Got \$data" }. Using the panel id and row and column indices provides faster data access than using panel, row, and column names. Panel ids and row and column indices that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel, row, and panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".	
<b>Example Usage</b>	<pre>load_package report project_open chiptrip load_report  # Set panel name and id set panel {Flow Settings} set id [get_report_panel_id \$panel] # Set row name set rname {Revision Name}  # Get row 2 - column 1 data get_report_panel_data -row 2 -col 1 -id \$id # Get row {Revision Name} - column 1 data get_report_panel_data -row_name \$pname -col 1 -id \$id # Get row {Revision Name} - column {Setting} data get_report_panel_data -row_name \$pname -col_name Setting -id \$id # If unsure the case of a row or column name, use glob-style pattern get_report_panel_data -row_name {[Rr]evision*} -col_name {[Ss]etting} -id \$id  unload_report project_close</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
	TCL_ERROR	1	ERROR: Found conflicting column options. Specify either the -col or -col_name option.
	TCL_ERROR	1	ERROR: Found conflicting row options. Specify either the -row or -row_name option.
	TCL_ERROR	1	ERROR: Table is empty. Add rows to the table before accessing its data.
	TCL_ERROR	1	ERROR: Illegal column name: <string>.
	TCL_ERROR	1	ERROR: Illegal column number: <string>. Specify a legal column number from <string> to <string>.
	TCL_ERROR	1	ERROR: Illegal row name: <string>.
	TCL_ERROR	1	ERROR: Illegal row number: <string>. Specify a legal row number from <string> to <string>.
	TCL_ERROR	1	ERROR: No column option specified.
	TCL_ERROR	1	ERROR: Can't retrieve data for row and column number. Specify a legal row and column number for the command get_report_panel_data.
	TCL_ERROR	1	ERROR: No row option specified.
	TCL_ERROR	1	ERROR: Can't access report panel. Use this command only for report panels with rows or columns.

### 3.1.26.9. get\_report\_panel\_id (::quartus::report)

The following table displays information for the get\_report\_panel\_id Tcl command:

Tcl Package and Version	Belongs to ::quartus::report on page 327	
Syntax	get_report_panel_id [-h   -help] [-long_help] <name>	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	<name>	Name of panel for which to get id
Description	Gets the id of a panel with the specified name. The panel id is a non-negative integer for an existing panel. If the specified panel cannot be found, the id is -1. Using the panel id provides faster data access than using the panel name. You can use the "get_report_panel_id" command to check for the existence of a panel.(Refer to the example under the "get_report_panel_id" command.) Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".	

*continued...*

<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Set panel name set panel {Fitter  Fitter Settings} # Get the panel id set id [get_report_panel_id \$panel]  if {\$id != -1} {     # If panel exists, add a row to it     add_row_to_table -id \$id {[New Field] Yes No}     # Save the changes to the report database     save_report_database } else {     # Otherwise print an error message     puts "Error: Table \$panel does not exist." }  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.26.10. get\_report\_panel\_names (::quartus::report)

The following table displays information for the `get_report_panel_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>get_report_panel_names [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns a list of panel names for the current report. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, each panel name returned by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path returned by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre> ## Load report database and write Timing Analyzer Summary ## panel to file in HTML format  load_package report project_open chiptrip load_report  # Set panel name set fmax_panel "Timing Analyzer Summary"  # Iterate through all the accessible panels foreach panel [get_report_panel_names] {     # If find the panel '*Timing Analyzer Summary',     # write its content to file fmax.htm     if {[string match "*\$fmax_panel" \$panel] == 1} {         write_report_panel -file fmax.htm -html \$panel         break     } }  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
TCL_ERROR	1	ERROR: Illegal report panel type.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.

### 3.1.26.11. get\_report\_panel\_row (::quartus::report)

The following table displays information for the get\_report\_panel\_row Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327	
<b>Syntax</b>	get_report_panel_row [-h   -help] [-long_help] [-id <table_id>] [-name <table_name>] [-row <row>] [-row_name <row_name>]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-id <table_id>	Id of panel from which to get data
	-name <table_name>	Name of panel from which to get data
	-row <row>	Row (or Y) coordinate
	-row_name <row_name>	Row (or Y) name
<b>Description</b>	Reports data of the specified panel row. Using the panel id and row index provides faster data access than using panel name and row name, respectively. Panel ids and row indices that you have cached may become outdated or invalid if the report gets unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel and row names support wildcards. The table of content portion of the UI Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the UI Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".	
<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Set panel name and id set panel {Flow Settings} set id [get_report_panel_id \$panel] # Set row name set rname {Revision Name}  # Get row {Revision Name} data (use panel id) get_report_panel_row -row_name \$name -id \$id # Get row 2 data (use panel id) get_report_panel_row -row 2 -id \$id # Get row 3 data (use panel name) get_report_panel_row -row 3 -name \$panel  ## Get the last row data set row_cnt [get_number_of_rows -id \$id] set last_rindex [expr \$row_cnt - 1] get_report_panel_row -row \$last_rindex -id \$id  unload_report project_close </pre>	

*continued...*

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
	TCL_ERROR	1	ERROR: Found conflicting row options. Specify either the -row or -row_name option.
	TCL_ERROR	1	ERROR: Table is empty. Add rows to the table before accessing its data.
	TCL_ERROR	1	ERROR: Illegal row name: <string>.
	TCL_ERROR	1	ERROR: Illegal row number: <string>. Specify a legal row number from <string> to <string>.
	TCL_ERROR	1	ERROR: No row option specified.
	TCL_ERROR	1	ERROR: Can't access report panel. Use this command only for report panels with rows or columns.

### 3.1.26.12. get\_report\_panel\_row\_index (::quartus::report)

The following table displays information for the `get_report_panel_row_index` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327	
<b>Syntax</b>	<code>get_report_panel_row_index [-h   -help] [-long_help] [-id &lt;table_id&gt;] [-name &lt;table_name&gt;] &lt;row_name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-id &lt;table_id&gt;</code>	Id of panel from which to get row index
	<code>-name &lt;table_name&gt;</code>	Name of panel from which to get row index
	<code>&lt;row_name&gt;</code>	Row name
<b>Description</b>	Gets the row index of the specified panel row name. Row name refers to the first element content of the specified row. The row index is a non-negative integer for an existing row. Row index is -1 if the row name is not found in the panel. Using the row index and panel id provides faster data access than using row name and panel name. Row indices and panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Row and panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis Analysis and Synthesis Summary".	
<b>Example Usage</b>	<pre>load_package report project_open chiptrip load_report  # Set panel name and id set panel {*Input Pins} set id      [get_report_panel_id \$panel]  # Set row name and index set rname  {[Cc]lk1} set rindex [get_report_panel_row_index -id \$id \$rname]</pre>	

*continued...*

```

# Get data out of the specified panel
set pc_str [get_report_panel_data -id $id -row 0 -col 1]
set pin_cnt [get_report_panel_data -id $id -row $rindex -col 1]
set iob_str [get_report_panel_data -id $id -row 0 -col 2]
set io_bank [get_report_panel_data -id $id -row $rindex -col 2]

# Output results
puts "$pc_str: $pin_cnt"
puts "$io_str: $io_bank"

unload_report
project_close

```

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.26.13. load\_report (::quartus::report)

The following table displays information for the `load_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>load_report [-h   -help] [-long_help] [-simulator]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-simulator	Option to load the Simulation Report. If this option isn't specified, the Compilation Report is loaded instead.	
<b>Description</b>	By default, loads the Compilation Report for the current revision or the specified revision name. If the <code>-simulator</code> option is specified, loads the Simulation Report instead. After the report is loaded or reloaded, the cached panel ids, and row and column indices may become outdated or invalid. Intel recommends that you update them before using them.		
<b>Example Usage</b>	<pre> # Load report package load_package report # Open chiptrip project project_open chiptrip # Load the current revision report load_report  # Set panel name and id set panel {Fitter  Fitter Summary} set id [get_report_panel_id \$panel]  # Get total registers set rname {Total [R]egisters} set rindex [get_report_panel_row_index -id \$id \$rname] set rname [get_report_panel_data -id \$id -row \$rindex -col 0] set data [get_report_panel_data -id \$id -row \$rindex -col 1] puts "\$rname: \$data"  # Get total pins set rname {Total [P]ins} set rindex [get_report_panel_row_index -id \$id \$rname] set rname [get_report_panel_data -id \$id -row \$rindex -col 0] set data [get_report_panel_data -id \$id -row \$rindex -col 1] puts "\$rname: \$data"  # Unload the report unload_report # Close the project project_close </pre>		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't load report for revision: <string>. A different, active revision already exists: <string>. To specify another active revision name, type set_current_revision <string>.

*continued...*

TCL_ERROR	1	ERROR: Can't find active revision. Specify an active revision name using set_current_revision <string>.
TCL_ERROR	1	ERROR: Can't find active revision. Specify an active revision name using set_current_revision <revision name>.
TCL_ERROR	1	ERROR: Revision name does not exist: <string>. Run Analysis and Synthesis (quartus_map) with the specified revision name before using this Tcl command.
TCL_ERROR	1	ERROR: Compilation database is newer than the last call to create_timing_netlist. Use the delete_timing_netlist Tcl command before using <string>.
TCL_ERROR	1	ERROR: You must open a project before you can use this command.
TCL_ERROR	1	ERROR: Can't load report data for revision name: <string>. Make sure the report database exists for the specified revision name.
TCL_ERROR	1	ERROR: Existing report was not unloaded. Ensure that your load_report commands have matching unload_report commands.

### 3.1.26.14. read\_xml\_report (::quartus::report)

The following table displays information for the read\_xml\_report Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327		
<b>Syntax</b>	read_xml_report [-h   -help] [-long_help] <filename>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<filename>	Name of XML Report Database File from which to read	
<b>Description</b>	Creates the current Compilation Report from the specified XML Report Database File (.xml).		
<b>Example Usage</b>	<pre># Set project name set project_name "chiptrip"  load_package report project_open \$project_name  # Read XML Report Database File (.xml) read_xml_report \$project_name.xml  load_report  # Get all the panel names puts {All Report Panel Names:} puts [get_report_panel_names]  unload_report project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Report read from XML Report Database File: <string>. No action is required.
	TCL_ERROR	1	ERROR: Can't read XML Report Database File: <string> . Make sure that the file exists and does not contain syntax errors.

### 3.1.26.15. refresh\_report\_window (::quartus::report)

The following table displays information for the `refresh_report_window` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327		
<b>Syntax</b>	<code>refresh_report_window [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Refresh the Report Window.		
<b>Example Usage</b>	<code>refresh_report_window</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.26.16. save\_report\_database (::quartus::report)

The following table displays information for the `save_report_database` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::report on page 327		
<b>Syntax</b>	<code>save_report_database [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Saves the report database, including any new report panel.		
<b>Example Usage</b>	<pre> load_package report project_open chiptrip load_report  # Set panel name and id set panel {Fitter  Fitter Settings} set id [get_report_panel_id \$panel]  # If panel exists, add a row to it. Otherwise, print an error message. if {\$id != -1} {     add_row_to_table -id \$id {{New Field} Yes No}     # Save the changes to the report database     save_report_database } else {     puts "Error: Table \$panel does not exist." }  unload_report project_close </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.
	TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.

### 3.1.26.17. unload\_report (::quartus::report)

The following table displays information for the `unload_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>unload_report [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Unloads the report for the current revision or the specified revision name. After the report is loaded or reloaded, the cached panel ids, and row and column indices, may become outdated or invalid. Intel recommends that you update them before using them.		
<b>Example Usage</b>	<pre># Load report package load_package report  # Open chiptrip project project_open chiptrip  # Load the current revision report load_report  # Set panel name and id set panel {Fitter  Fitter Summary} set id    [get_report_panel_id \$panel]  # Get total registers set rname {Total [Rr]egisters} set rindex [get_report_panel_row_index -id \$id \$rname] set rname [get_report_panel_data -id \$id -row \$rindex -col 0] set data  [get_report_panel_data -id \$id -row \$rindex -col 1] puts "\$rname: \$data"  # Get total pins set rname {Total [Pp]ins} set rindex [get_report_panel_row_index -id \$id \$rname] set rname [get_report_panel_data -id \$id -row \$rindex -col 0] set data  [get_report_panel_data -id \$id -row \$rindex -col 1] puts "\$rname: \$data"  # Unload the report unload_report # Close the project project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't load report for revision: <string>. A different, active revision already exists: <string>. To specify another active revision name, type set_current_revision <string>.
	TCL_ERROR	1	ERROR: Can't find active revision. Specify an active revision name using set_current_revision <string>.
	TCL_ERROR	1	ERROR: Can't find active revision. Specify an active revision name using set_current_revision <revision name>.
	TCL_ERROR	1	ERROR: You must open a project before you can use this command.

### 3.1.26.18. write\_report\_panel (::quartus::report)

The following table displays information for the `write_report_panel` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327
<b>Syntax</b>	<code>write_report_panel [-h   -help] [-long_help] -file &lt;output file name&gt; [-html] [-id &lt;table_id&gt;] [-name &lt;table_name&gt;] [-xml] [&lt;name&gt;]</code>
<i>continued...</i>	

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-file &lt;output file name&gt;</code>	Name of output file to be generated	
	<code>-html</code>	Option to generate output file in HTML format	
	<code>-id &lt;table_id&gt;</code>	id of panel from which to get data	
	<code>-name &lt;table_name&gt;</code>	Name of panel from which to get data	
	<code>-xml</code>	Option to generate output file in XML format	
	<code>&lt;name&gt;</code>	Name of panel from which to get data	
<b>Description</b>	Writes data from the specified report panel to the specified output file. If the "-html" option is specified, the output file is generated in HTML format. If the "-xml" option is specified, the output file is generated in XML format. Otherwise, the output file is generated in ASCII format. Using the panel id provides faster data access than using the panel name. Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands. Panel names support wildcards. The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis and Synthesis  Summary". However, the corresponding full path used by this Tcl command is "Analysis and Synthesis  Analysis and Synthesis Summary".		
<b>Example Usage</b>	<pre>## Load report database and write Timing Analyzer Summary ## panel to file in HTML format  load_package report project_open chiptrip load_report  # Set panel name and id set panel "*Timing Analyzer Summary" set id    [get_report_panel_id \$panel]  # If the specified panel exists, write it to # fmax.htm and fmax.xml. # Otherwise, print out an error message if {\$id != -1} {     write_report_panel -file fmax.htm -html -id \$id     write_report_panel -file fmax.xml -xml -id \$id } else {     puts "Error: report panel could not be found." }  unload_report project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Successfully generated HTML-Format Report File: < <i>string</i> >
	TCL_OK	0	INFO: Successfully generated output file: < <i>string</i> >
	TCL_OK	0	INFO: Successfully generated XML-Format Report File: < <i>string</i> >
	TCL_OK	0	INFO: Report automatically reloaded because it was not up-to-date after use of Tcl command execute_flow or execute_module (which belong to ::quartus::flow package). No action is required.
	TCL_ERROR	1	ERROR: Can't create or overwrite file: < <i>string</i> >. Specify a file name that has write permission.

*continued...*

TCL_ERROR	1	ERROR: Can't find the top-level panel. Make sure the loaded report is not empty. You can run a successful compilation to rebuild the report.
TCL_ERROR	1	ERROR: Can't find panel: <string>. Specify an existing report panel name.
TCL_ERROR	1	ERROR: Report not loaded for revision name: <string>. Type load_report to load the report.

### 3.1.26.19. write\_xml\_report (::quartus::report)

The following table displays information for the `write_xml_report` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::report</a> on page 327		
<b>Syntax</b>	<code>write_xml_report [-h   -help] [-long_help] &lt;filename&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<filename>	Name of XML Report Database File to which to write	
<b>Description</b>	Writes the current Compilation Report or Simulation Report to the specified XML Report Database File (.xml).		
<b>Example Usage</b>	<pre>load_package report set project_name "chiptrip" project_open \$project_name  ## Create XML Report Database File (.xml) load_report file delete -force report.xml puts [write_xml_report report.xml] unload_report  # Close project project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Report written to XML Report Database File: <string>. No action is required.
	TCL_ERROR	1	ERROR: Can't write XML Report Database File: <string> -- report does not exist. Make sure that an existing report is loaded. Type load_report -h for more information.

### 3.1.27. ::quartus::sdc

The following table displays information for the `::quartus::sdc` Tcl package:

<b>Tcl Package and Version</b>	::quartus::sdc 1.5		
<b>Description</b>	Synopsys Design Constraint (SDC) format is used to specify the design intent, including the timing and area constraints of the design. The Timing Analyzer only implements the set of SDC commands required to specify the timing constraints of the design. For area constraints, the QSF file should be used. This package implements the SDC Spec Version 1.5 (June 2005). Any command in this package can be specified in a Timing Analyzer SDC file.		
<b>Availability</b>	This package is loaded by default in the following executable: <code>quartus_sta</code>		

*continued...*

	This package is available for loading in the following executable: <code>quartus_fit</code>
<b>Tcl Commands</b>	<a href="#">all_clocks</a> (::quartus::sdc) on page 347 <a href="#">all_inputs</a> (::quartus::sdc) on page 348 <a href="#">all_outputs</a> (::quartus::sdc) on page 348 <a href="#">all_registers</a> (::quartus::sdc) on page 349 <a href="#">create_clock</a> (::quartus::sdc) on page 349 <a href="#">create_generated_clock</a> (::quartus::sdc) on page 350 <a href="#">derive_clocks</a> (::quartus::sdc) on page 352 <a href="#">get_cells</a> (::quartus::sdc) on page 352 <a href="#">get_clocks</a> (::quartus::sdc) on page 353 <a href="#">get_nets</a> (::quartus::sdc) on page 354 <a href="#">get_pins</a> (::quartus::sdc) on page 355 <a href="#">get_ports</a> (::quartus::sdc) on page 356 <a href="#">remove_clock_groups</a> (::quartus::sdc) on page 357 <a href="#">remove_clock_latency</a> (::quartus::sdc) on page 357 <a href="#">remove_clock_uncertainty</a> (::quartus::sdc) on page 358 <a href="#">remove_disable_timing</a> (::quartus::sdc) on page 359 <a href="#">remove_input_delay</a> (::quartus::sdc) on page 359 <a href="#">remove_output_delay</a> (::quartus::sdc) on page 360 <a href="#">reset_design</a> (::quartus::sdc) on page 361 <a href="#">set_clock_groups</a> (::quartus::sdc) on page 361 <a href="#">set_clock_latency</a> (::quartus::sdc) on page 362 <a href="#">set_clock_uncertainty</a> (::quartus::sdc) on page 363 <a href="#">set_disable_timing</a> (::quartus::sdc) on page 364 <a href="#">set_false_path</a> (::quartus::sdc) on page 365 <a href="#">set_input_delay</a> (::quartus::sdc) on page 366 <a href="#">set_input_transition</a> (::quartus::sdc) on page 368 <a href="#">set_max_delay</a> (::quartus::sdc) on page 368 <a href="#">set_max_time_borrow</a> (::quartus::sdc) on page 370 <a href="#">set_min_delay</a> (::quartus::sdc) on page 371 <a href="#">set_multicycle_path</a> (::quartus::sdc) on page 372 <a href="#">set_output_delay</a> (::quartus::sdc) on page 374

### 3.1.27.1. all\_clocks (::quartus::sdc)

The following table displays information for the `all_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>all_clocks [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns a collection of all clocks in the design.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist foreach_in_collection clk [all_clocks] {     puts [get_clock_info -name \$clk] } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.2. all\_inputs (::quartus::sdc)

The following table displays information for the all\_inputs Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	all_inputs [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns a collection of all input ports in the design.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist foreach_in_collection in [all_inputs] {     puts [get_port_info -name \$in] } set_input_delay -clock clock1 2.0 [all_inputs] delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.3. all\_outputs (::quartus::sdc)

The following table displays information for the all\_outputs Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	all_outputs [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns a collection of all output ports in the design.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist foreach_in_collection out [all_outputs] {     puts [get_port_info -name \$out] } set_output_delay -clock clock1 2.0 [all_outputs] delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.4. all\_registers (::quartus::sdc)

The following table displays information for the `all_registers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>all_registers [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns a collection of all registers in the design.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist foreach_in_collection reg [all_registers] {     puts [get_register_info -name \$reg] } report_timing -from [all_registers] -to [all_registers] delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.5. create\_clock (::quartus::sdc)

The following table displays information for the `create_clock` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>create_clock [-h   -help] [-long_help] [-add] [-name &lt;clock_name&gt;] -period &lt;value&gt; [-waveform &lt;edge_list&gt;] [ &lt;targets&gt; ]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-add</code>	Adds clock to a node with an existing clock	
	<code>-name &lt;clock_name&gt;</code>	Clock name of the created clock	
	<code>-period &lt;value&gt;</code>	Speed of the clock in terms of clock period	
	<code>-waveform &lt;edge_list&gt;</code>	List of edge values	
	<code>&lt;targets&gt;</code>	List or collection of targets	
<b>Description</b>	Defines a clock. If the <code>-name</code> option is not used, the clock name is the same as the first target in the list or collection. The clock name is used to refer to the clock in other commands. The <code>-period</code> option specifies the clock period. It is also possible to use this option to specify a frequency to define the clock period. This can be done by using <code>-period</code> option followed by either <code>&lt;frequency&gt;MHz</code> or <code>"&lt;frequency&gt; MHz"</code> . However, this is a Timing Analyzer-only extension and makes the SDC syntax non-standard. The <code>-waveform</code> option specifies the rising and falling edges (duty cycle) of the clock, and is specified as a list of two time values: the first rising edge and the next falling edge. The rising edge must be within the range <code>[0, period]</code> . The falling edge must be within one clock period of the rising edge. The waveform defaults to <code>(0, period/2)</code> . If a clock with the same name is already assigned to a given target, the <code>create_clock</code> command will overwrite the existing clock. If a clock with a different name exists on the given target, the <code>create_clock</code> command will be ignored unless the <code>-add</code> option is used. The <code>-add</code> option can be used to assign multiple clocks to a pin or port. If the target of the clock is internal (i.e. not an input port), the source latency is zero by default. If a clock is on a path after another clock, then it blocks or overwrites the previous clock from that point forward. The		

*continued...*

	<p>value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.</p>									
<b>Example Usage</b>	<pre># Create a simple 10ns with clock with a 60% duty cycle create_clock -period 10 -waveform {0 6} -name clk [get_ports clk]  # Create a clock with a falling edge at 2ns, rising edge at 8ns, # falling at 12ns, etc. create_clock -period 10 -waveform {8 12} -name clk [get_ports clk]  # Assign two clocks to an input port that are switched externally create_clock -period 10 -name clk100Mhz [get_ports clk] create_clock -period 6.667 -name clk150Mhz -add [get_ports clk]  # Two ways to use MHz to define clock period (Timing Analyzer only) create_clock -period 250MHz -name clk250MHz [get_ports clk] create_clock -period "250 MHz" -name clk250MHz [get_ports clk]</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.27.6. create\_generated\_clock (::quartus::sdc)

The following table displays information for the `create_generated_clock` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	<code>create_generated_clock [-h   -help] [-long_help] [-add] [-divide_by &lt;factor&gt;] [-duty_cycle &lt;percent&gt;] [-edge_shift &lt;shift_list&gt;] [-edges &lt;edge_list&gt;] [-invert] [-master_clock &lt;clock&gt;] [-multiply_by &lt;factor&gt;] [-name &lt;clock_name&gt;] [-offset &lt;time&gt;] [-phase &lt;degrees&gt;] [-source &lt;clock_source&gt;] [ &lt;targets&gt; ]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-add	Add clock to existing clock node
	-divide_by <factor>	Division factor
	-duty_cycle <percent>	Specifies the duty cycle as a percentage of the clock period--accepts floating point values
	-edge_shift <shift_list>	List of edge shifts
	-edges <edge_list>	List of edge values
	-invert	Invert the clock waveform
	-master_clock <clock>	Specifies clock of the source node
	-multiply_by <factor>	Multiplication factor
	-name <clock_name>	Name of generated clock
	-offset <time>	Specifies the offset as an absolute time shift
	-phase <degrees>	Specifies the phase shift in degrees
	-source <clock_source>	Source node for the generated clock
<targets>		List or collection of targets

*continued...*

<b>Description</b>	<p>Defines an internally generated clock. If -name is not specified, the clock name is the same as the first target in the list or collection. The clock name is used to refer to the clock in other commands. If a clock with the same name is already assigned to a given target, the <code>create_generated_clock</code> command overwrites the existing clock. If a clock with a different name exists on the given target, the <code>create_generated_clock</code> command is ignored unless the -add option is used. The -add option can be used to assign multiple clocks to a pin or port, and is recommended be used with -master_clock option. The source of the generated clock, specified by -source, is a port, pin, register, or net in the design. All waveform modifications are relative to this point. If more than one clock feeds the source node, the -master_clock option must be used to specify which clock to modify. The source latency of the generated clock is based on the clock network of the generated clock, and not the clock network of the node specified using -source. This latency is added to any source latency of the master clock. If no target is specified, the clock is treated as a virtual clock. In that case, the source latency of the generated clock will be equal to the source latency of the master clock, plus any added latency specified with <code>set_clock_latency</code>. The -divide_by, -multiply_by, -invert, -duty_cycle, -edges, and -edge_shift options modify the waveform relative to the waveform at the source node. Clock division and multiplication, using -divide_by and -multiply_by, is performed relative to the first rising edge. Clock division is based on edges in the master clock waveform, and scaled if the division is an odd number. Use the -duty_cycle option to specify the new duty cycle for clock multiplication. Use the -phase option to specify any phase shift relative to the new clock period. Use the -offset option to specify an arbitrary offset or time shift. Use the -invert option to invert the generated waveform. The -phase and -duty_cycle options may be specified as a decimal value (e.g. 22.5) or as a ratio of two numbers (e.g. 45/2). The latter form may improve Timing Analyzer accuracy when detecting relationships between related clocks. Clock generation can also be specified with the -edges and -edge_shift options. The -edges option accepts a list of three numbers specifying the master clock edges to use for the first rising edge, the next falling edge, and next rising edge. Edges of the master clock are labeled according to the first rising edge (1), next falling edge (2), next rising edge (3), etc. For example, a basic clock divider can be specified equivalently with -divide_by 2 or -edges {1 3 5}. The -edge_shift option accepts a list of three time values, the amount to shift each of the three edges. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.</p>
<b>Example Usage</b>	<pre># Create a clock and a divide-by-2 generated clock create_clock -period 10 [get_ports clk] create_generated_clock -divide_by 2 -source [get_ports clk] -name clkdiv [get_registers clkdiv]  # An equivalent generated clock create_generated_clock -edges {1 3 5} -source [get_ports clk] -name clkdiv [get_registers clkdiv]  # Specify a clock multiplier with a 60% duty cycle create_generated_clock -multiply_by 2 -source [get_ports clk] -duty_cycle 60 [get_pins clkmult combout]  # Specify an inverted divide-by-2 clock relative to the output of the source clock create_generated_clock -divide_by 2 -invert -source [get_ports clk] -name nclkdiv [get_registers clkdiv]  # Specify a divide-by-2 clock with a 90-degree phase shift create_generated_clock -divide_by 2 -phase 90 -source [get_ports clk] -name clkdiv [get_registers clkdiv]  # Create a divide-by-2 generated clock generated off the falling edge of the source clock create_generated_clock -edges {2 4 6} -source [get_ports clk] -name clkfall_div [get_registers clkfall_div]  # Assign two clocks to an input port that are switched externally, # along with an internal clock divider. create_clock -period 10 -name clk100Mhz [get_ports clk] create_clock -period 6.667 -name clk150Mhz -add [get_ports clk] create_generated_clock -divide_by 2 -name clk50Mhz -source [get_ports clk] -master_clock clk100Mhz -add [get_registers clkdiv] create_generated_clock -divide_by 2 -name clk75Mhz -source [get_ports clk] -master_clock clk150Mhz -add [get_registers clkdiv]  # Create a virtual clock, and two generated clocks that derive from it. # This makes the generated clocks related, so crossings between them are not asynchronous. create_clock -period 10 -name virtual_base create_generated_clock -master_clock virtual_base -divide_by 2 [get_ports clka] create_generated_clock -master_clock virtual_base -divide_by 4 [get_ports clkb]</pre>

**continued...**

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.7. derive\_clocks (::quartus::sdc)

The following table displays information for the derive\_clocks Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	derive_clocks [-h   -help] [-long_help] -period <period_value> [-waveform <edge_list> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-period <period_value>	Speed of the default clock in terms of clock period	
	-waveform <edge_list>	List of edge values	
<b>Description</b>	Creates a clock on sources of clock pins in the design that do not already have at least one clock sourcing the clock pin. This command is equivalent to calling create_clock on each clock source in the design that does not already have a clock assigned to it. See the help for create_clock for more information. Intel does not recommend using this command during final sign-off analysis of a design. derive_clocks should only be used early in the design phase when the clocks are not completely known. When possible, create_clock and create_generated_clock should be used instead.		
<b>Example Usage</b>	# Automatically create a 10ns, 60% duty cycle clock on all # unconstrained clock sources derive_clocks -period 10 -waveform {0 6}		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.27.8. get\_cells (::quartus::sdc)

The following table displays information for the get\_cells Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	get_cells [-h   -help] [-long_help] [-compatibility_mode] [-hierarchical] [-no_duplicates] [-nocase] [-nowarn] [ <filter> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-compatibility_mode	Use simple Tcl matching	
	-hierarchical	Specifies use of a hierarchical searching method	
	-no_duplicates	Do not match duplicated cell names	
	-nocase	Specifies case insensitive node name matching	
	-nowarn	Do not issue warning messages about unmatched patterns	
	<filter>	Valid destinations (string patterns are matched using Tcl string matching)	

*continued...*

<b>Description</b>	Returns a collection of cells in the design. All cell names in the collection match the specified pattern. Wildcards can be used to select multiple cells at once. There are three Tcl string matching schemes available with this command: the default method, the -hierarchical option, and the -compatibility_mode option. When you use the default matching scheme, use pipe characters to separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. When this matching scheme is enabled, the specified pattern is matched against absolute cell names: the names that include the entire hierarchical path. A full cell name can contain multiple pipe characters in it to reflect the hierarchy. All hierarchy levels in the pattern are matched level by level. Any included wildcards refer to only one hierarchical level. For example, "*" and "* *" produce different collections since they refer to the highest hierarchical level and second highest hierarchical level respectively. When using the -hierarchical matching scheme, pipe characters are treated as special characters and are taken into account when string matching with wildcards is performed. This matching scheme forces the search to proceed recursively down the hierarchy. The specified pattern is matched against the relative cell names: the immediate names that do not include any of the hierarchy information. Note that a short cell name cannot contain pipe characters in it. Any included wildcards are expanded to match the relative cell names. The -compatibility_mode matching scheme uses simple Tcl string matching on full, absolute cell names. Pipe characters are not treated as special characters when used with wildcards. The default matching scheme returns cells whose names match the specified filter and also cells automatically generated by the Quartus II software from these cells). Use -no_duplicates option to not include duplicated cells. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.									
<b>Example Usage</b>	<pre># Find a cell called "reg" using case insensitive search get_cells -nocase reg # Create a collection of all cells whose names start with "reg" get_cells reg* # Create a collection of all cells on the highest hierarchical level set mycollection [get_cells *] # Create a collection of all cells in the design # Output cell names. foreach_in_collection cell \$mycollection {     puts [get_cell_info -name \$cell] } set fullcollection [get_cells -hierarchical *] # Output cell IDs and names. foreach_in_collection cell \$fullcollection {     puts -newline \$cell     puts -newline ":" "     puts [get_cell_info -name \$cell] }</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th> <th><b>Code</b></th> <th><b>String Return</b></th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.27.9. get\_clocks (::quartus::sdc)

The following table displays information for the get\_clocks Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	get_clocks [-h   -help] [-long_help] [-include_generated_clocks] [-nocase] [-nowarn] [-of_objects <object_collection>] [<filter>]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-include_generated_clocks	Includes generated clocks derived from the matched clocks
	-nocase	Specifies the matching of node names to be case-insensitive
	-nowarn	Do not issue warning messages when querying for clocks

*continued...*

	<table border="1"> <tr> <td><code>-of_objects &lt;object_collection&gt;</code></td><td>Returns all clocks that target (defined on) or drive (determine data frequency) the nodes in the collection.</td></tr> <tr> <td><code>&lt;filter&gt;</code></td><td>Valid destinations (string patterns are matched using Tcl string matching)</td></tr> </table>	<code>-of_objects &lt;object_collection&gt;</code>	Returns all clocks that target (defined on) or drive (determine data frequency) the nodes in the collection.	<code>&lt;filter&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)					
<code>-of_objects &lt;object_collection&gt;</code>	Returns all clocks that target (defined on) or drive (determine data frequency) the nodes in the collection.									
<code>&lt;filter&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)									
<b>Description</b>	Returns a collection of clocks in the design. Use a clock collection as the -from/to argument of a command (such as <code>set_multicycle_path</code> ) to refer to all nodes driven by the clocks in the collection. # The following multicycle constraint applies to all paths ending at registers # driven by clk <code>set_multicycle_path -to [get_clocks clk] 2</code> If a filter, which is a Tcl list of wildcards and must follow standard Tcl or Timing Analyzer-extension substitution rules, is specified, then <code>get_clocks</code> returns all clocks whose names match the filter. See the help for <code>use_timing_analyzer_styleEscaping</code> for filter rules. If the <code>-of_objects</code> option is used, then a collection of registers, ports, pins, or cells must be provided. <code>get_clocks</code> returns a collection of all the clocks that target these nodes, or if these nodes are not clock targets, all the clocks that drive these nodes. This option cannot be used along with the clock name filter. Refer to the long help for examples of using the <code>-of_objects</code> option.									
<b>Example Usage</b>	<pre># get clocks that begin with 'c' or 'C', and print out their names and periods: set clocks [get_clocks c* -nocase] foreach_in_collection clk \$clocks {     set name [get_clock_info -name \$clk]     set period [get_clock_info -period \$clk]     puts "\$name: \$period" }  # getting the clock that targets a port, and its generated clock: create_clock -name my_clock -period 10.000 [get_ports CLK_100] create_generated_clock -name my_gen_clock -divide_by 2 -source [get_ports CLK_100] [get_registers clk_div_reg] get_clocks -nowarn -of_objects [get_ports CLK_100] -include_generated_clocks  # When you use the -nowarn option with the -of_objects option, your filter should be a node collection # generated by another get_* collection command as shown in the examples. # # When you use the -of_objects option, the Timing Analyzer may emit many clock-related warning messages. These warnings # can typically be ignored using the -nowarn option if SDC files read later in the flow create more clocks. # However, the -nowarn option also suppresses warnings caused by unmatched bareword filters. To only see warnings # due to unmatched filters, use the -nowarn option; however, instead of passing in a bareword filter, pass in a # node collection generated from another get_* collection command.  # display the name of the clocks that drive registers beginning with 'reg_': foreach_in_collection clk_id [get_clocks -nowarn -of_objects [get_registers reg_*]] {     puts [get_clock_info -name \$clk_id] }</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.								

### 3.1.27.10. `get_nets` (::quartus::sdc)

The following table displays information for the `get_nets` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	<code>get_nets [-h   -help] [-long_help] [-no_duplicates] [-nocase] [-nowarn] [&lt;filter&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-no_duplicates</code>	Do not match duplicated net names
	<code>-nocase</code>	Specifies case-insensitive node name matching

*continued...*

	<code>-nowarn</code>	Do not issue warning messages about unmatched patterns	
	<code>&lt;filter&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)	
<b>Description</b>	Returns a collection of nets in the design. All net names in the collection match the specified pattern. Wildcards can be used to select multiple nets at once. The default matching scheme returns nets whose names match the specified filter and nets that are automatically generated by the Quartus Prime software from these nets. Use the <code>-no_duplicates</code> option to exclude duplicated nets. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <code>use_timing_analyzer_styleEscaping</code> command for details.		
<b>Example Usage</b>	<pre># Find a net called "reg" using case insensitive search get_nets -nocase reg # Create a collection of all nets whose names start with "reg" get_nets reg* # Create a collection of all nets in the design set mycollection [get_nets *] # Output net names. foreach_in_collection net \$mycollection {     puts [get_net_info -name \$net] }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.27.11. get\_pins (::quartus::sdc)

The following table displays information for the `get_pins` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc</a> on page 346	
<b>Syntax</b>	<code>get_pins [-h   -help] [-long_help] [-compatibility_mode] [-hierarchical] [-no_duplicates] [-nocase] [-nowarn] [ &lt;filter&gt; ]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-compatibility_mode</code>	Use simple Tcl matching
	<code>-hierarchical</code>	Specifies use of a hierarchical searching method
	<code>-no_duplicates</code>	Do not match duplicated pin names
	<code>-nocase</code>	Specifies case-insensitive node name matching
	<code>-nowarn</code>	Do not issue warning messages about unmatched patterns
	<code>&lt;filter&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
<b>Description</b>	Returns a collection of pins in the design. All pin names in the collection match the specified pattern. Wildcards can be used to select multiple pins at once. There are three Tcl string matching schemes available with this command: the default method, the <code>-hierarchical</code> option, and the <code>-compatibility_mode</code> option. By default, pipe characters are used to separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. When the default matching scheme is enabled, the specified pattern is matched against absolute pin names: the names that include the entire hierarchical path. All hierarchy levels in the pattern are matched level by level. Pin names of the form <code>&lt;absolute full cell name&gt; &lt;pin suffix&gt;</code> are used for matching. Note that a full cell name can contain multiple pipe characters in it to reflect the hierarchy. Any included wildcards refer to only one hierarchical level. For example, <code>"* *"</code> and <code>"* * *"</code> produce different collections since they refer to the highest hierarchical level and second highest hierarchical level respectively. When using the <code>-hierarchical</code> matching scheme, pipe characters are treated as special characters and are taken into account when string	

*continued...*

	<p>matching with wildcards is performed. This matching scheme forces the search to proceed recursively through the hierarchy. The specified pattern is matched against the relative pin names: the immediate names that do not include any of the hierarchy information. Pin names of the form &lt;relative short cell name&gt; &lt;pin suffix&gt; are used for matching. Note that a short cell name cannot contain pipe characters. Any included wildcards are expanded to match the relative pin names. For example, "*" and "* *" match exactly the same pins since the former is expanded into the latter. The -compatibility_mode matching scheme uses simple Tcl string matching on full, absolute cell names. Pipe characters are not treated as special characters when used with wildcards. The default matching scheme returns not only pins whose names match the specified filter, but also pins duplicated from these pins (refers to pins automatically generated by Quartus from the pins). Use -no_duplicates option to not include duplicated pins. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.</p>									
<b>Example Usage</b>	<pre># Get regout pin of "reg" cell get_pins -nocase reg regout # Create a collection of all pins of "reg" cell get_pins reg * # Create a collection of all pins on the highest hierarchical level set mycollection [get_pins *] # Output pin names. foreach_in_collection pin \$mycollection {     puts [get_pin_info -name \$pin] } # Create a collection of all pins in the design set fullcollection [get_pins -hierarchical *] # Output pin IDs and names. foreach_in_collection pin \$fullcollection {     puts -nonewline \$pin     puts -nonewline ":"     puts [get_pin_info -name \$pin] }</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.27.12. get\_ports (::quartus::sdc)

The following table displays information for the get\_ports Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	get_ports [-h   -help] [-long_help] [-nocase] [-nowarn] [ <filter> ]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-nocase	Specifies case-insensitive node name matching
	-nowarn	Do not issue warning messages about unmatched patterns
	<filter>	Valid destinations (string patterns are matched using Tcl string matching)
<b>Description</b>	Returns a collection of ports (design inputs and outputs) in the design. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.	
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  # Get all ports starting with "In". set ports [get_ports In*] foreach_in_collection port \$ports {     puts [get_port_info -name \$port] }</pre>	

*continued...*

	delete_timing_netlist project_close		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.13. remove\_clock\_groups (::quartus::sdc)

The following table displays information for the `remove_clock_groups` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>remove_clock_groups [-h   -help] [-long_help] -all</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-all		Specify remove all clock group settings
<b>Description</b>	Remove all clock group assignments. This command removes any clock groups that have been previously set. There is no way to remove specific groups.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist create_clock -period 10.000 -name clkA [get_ports sysclk[0]] create_clock -period 10.000 -name clkB [get_ports sysclk[1]]  # Set clkA and clkB to be mutually exclusive clocks. set_clock_groups -exclusive -group {clkA} -group {clkB} set_clock_groups -exclusive -group {clkC} -group {clkD}  # Remove clock groups A, B, C, and D. Result is that there # are no longer any mutually exclusive clocks. remove_clock_groups -all</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.27.14. remove\_clock\_latency (::quartus::sdc)

The following table displays information for the `remove_clock_latency` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>remove_clock_latency [-h   -help] [-long_help] -source &lt;targets&gt;</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-source		Specifies the source clock latency
	<targets>		Valid destinations (string patterns are matched using Tcl string matching)

*continued...*

<b>Description</b>	Removes clock latency for a given clock or clock target. There are two types of latency: network and source. Network latency is the clock network delay between the clock and register clock pins. Source latency is the clock network delay between the clock and its source (e.g., a system clock or a base clock of a generated clock). The Timing Analyzer automatically computes network latencies for all register and generated clocks. Overriding clock network latencies is not supported by the Timing Analyzer. Therefore, the -source option must always be specified. Remove_clock_latency requires this option as well. You can apply clock latency to a clock, which affects all targets of the clock, or to a specific clock target. Therefore, you can remove clock latency from a collection of clocks, or from a collection of target nodes. remove_clock_latency removes all latencies from a clock or node, so removing a node's clock latency with respect to a particular clock, or removing only latencies with particular conditions is not supported. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.									
<b>Example Usage</b>	<pre>create_clock -name SYSCLK -period 10.000 [get_ports inclk] create_generated_clock -name OUTCLK -divide_by 1 -source [get_ports inclk] [get_ports outclk] create_generated_clock -name FDBKCLK -divide_by 1 -source [get_ports outclk] [get_ports fdbkclk]  # Apply a simple 2.000 ns source latency to the system clock. set_clock_latency -source 2.000 [get_clocks SYSCLK]  # Specify feedback clock latencies between output port outclk # and the output port fdbkclk. set_clock_latency -source -late -rise 0.800 [get_clocks FDBKCLK] set_clock_latency -source -late -fall 0.750 [get_clocks FDBKCLK] set_clock_latency -source -early -rise 0.500 [get_clocks FDBKCLK] set_clock_latency -source -early -fall 0.460 [get_clocks FDBKCLK]  # Remove all clock latency from FDBKCLK remove_clock_latency -source [get_clocks FDBKCLK]</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.27.15. remove\_clock\_uncertainty (::quartus::sdc)

The following table displays information for the remove\_clock\_uncertainty Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	remove_clock_uncertainty [-h   -help] [-long_help] -from <from_clock> -to <to_clock>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-from <from_clock>	Valid destinations (string patterns are matched using Tcl string matching)
	-to <to_clock>	Valid destinations (string patterns are matched using Tcl string matching)
<b>Description</b>	Removes clock uncertainty from a collection of clocks to a collection of clocks. The source and destination clocks can be any arbitrary collection of clocks. This command removes all uncertainty between two clocks. If there does not exist uncertainty between two clocks specified in remove_clock_uncertainty, the command does nothing for those two clocks but continues to attempt to remove uncertainty between other clocks specified. The values of the -from and -to options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.	

*continued...*

<b>Example Usage</b>	<pre>set_clock_uncertainty -setup -rise_from {clk1 clk2} -fall_to {clk3 clk4} 200ps set_clock_uncertainty -from {clk5 clk6} -to {clk7 clk8} 300ps remove_clock_uncertainty -from {clk3 clk5} -to {clk4 clk7}</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.27.16. remove\_disable\_timing (::quartus::sdc)

The following table displays information for the `remove_disable_timing` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>remove_disable_timing [-h   -help] [-long_help] [-from &lt;name&gt;] [-to &lt;name&gt;] &lt;cells&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-from <name>	Valid source pin suffix	
	-to <name>	Valid destination pin suffix	
	<cells>	List of cells	
<b>Description</b>	Adds a previously disabled edge (arc) back to a given cell(s). If no -from/-to value is specified, the missing value is substituted by a "*". The values of the -from and -to are valid pin suffixes. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.		
<b>Example Usage</b>	<pre>remove_disable_timing -from datain -to combout A B remove_disable_timing -from carryin *</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.27.17. remove\_input\_delay (::quartus::sdc)

The following table displays information for the `remove_input_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>remove_input_delay [-h   -help] [-long_help] [-blackbox] &lt;targets&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-blackbox	Removes an input delay that was assigned to a partition boundary port.	
	<targets>	Collection or list of input ports	
<b>Description</b>	Removes input delay from a port. For each input port specified, removes all input delays for that port. This means that rise, fall, max, and min delays for each clock and reference pin on the input port are all removed. The value of the targets is either a collection or a Tcl list of wildcards used to create a		
	<i>continued...</i>		

	collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.		
<b>Example Usage</b>	<pre># Simple input delay with the same value for min/max and rise/fall set_input_delay -clock clk 1.5 [get_ports {in1 in2}] set_input_delay -clock clk2 1.5 [get_ports {in1 in2}] set_input_delay -clock clk 1.6 [get_ports {in3 in4}]  # Remove input delay on ports in1 and in4, # for all flags and reference ports and flags remove_input_delay [get_ports {in1 in4}]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Incorrect collection type. Expected a collection of type <string>.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.18. remove\_output\_delay (::quartus::sdc)

The following table displays information for the remove\_output\_delay Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	remove_output_delay [-h   -help] [-long_help] [-blackbox] <targets>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-blackbox	Removes an output delay that was assigned to a partition boundary port.	
	<targets>	Collection or list of output ports	
<b>Description</b>	Removes output delay from a port. For each output port specified, removes all output delays for that port. Rise, fall, max, and min delays for each clock and reference pin on the output port are all removed. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.		
<b>Example Usage</b>	<pre># Simple output delay with the same value for min/max and rise/fall set_output_delay -clock clk 1.5 [get_ports {out1 out2}] set_output_delay -clock clk2 1.5 [get_ports {out1 out2}] set_output_delay -clock clk 1.6 [get_ports {out3 out4}]  # Remove input delay on ports out1 and out4, # for all flags and reference ports and flags remove_output_delay [get_ports {out1 out4}]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Incorrect collection type. Expected a collection of type <string>.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.19. reset\_design (::quartus::sdc)

The following table displays information for the `reset_design` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>reset_design [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Removes all assignments from the design. This includes clocks, generated clocks, derived clocks, input delays, output delays, clock latency, clock uncertainty, clock groups, false paths, multicycle paths, min delays, and max delays. After <code>reset_design</code> is called, the design should be in the same state as it would be if <code>create_timing_netlist</code> was just called.		
<b>Example Usage</b>	<pre># Constrain design create_clock -name clk -period 4.000 -waveform { 0.000 2.000 } [get_ports clk] set_input_delay -clock clk 2.1.5 [get_ports in*] set_output_delay -clock clk 1.6 [get_ports out*] set_false_path -from [get_keepers in] -through [get_nets r1] -to [get_keepers out]  # Reset the design to the state that it was in before any constraints were entered reset_design</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.27.20. set\_clock\_groups (::quartus::sdc)

The following table displays information for the `set_clock_groups` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<code>set_clock_groups [-h   -help] [-long_help] [-asynchronous] [-exclusive] -group &lt;names&gt; [-logically_exclusive] [-physically_exclusive]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-asynchronous</code>	Specify mutually exclusive clocks (such as groups of primary clocks)	
	<code>-exclusive</code>	Specify mutually exclusive clocks (an alias for the -logically_exclusive option). Exists for backwards compatibility.	
	<code>-group &lt;names&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)	
	<code>-logically_exclusive</code>	Specify logically exclusive clocks (meaning they are not actively used at the same time)	
	<code>-physically_exclusive</code>	Specify physically exclusive clocks (meaning they are not physically present at the same time)	
<b>Description</b>	Clock groups provide a quick and convenient way to specify which clocks are not related. Asynchronous clocks are those that are completely unrelated (e.g., have different ideal clock sources). Logically exclusive clocks are not actively used in the design at the same time (e.g., multiplexed clocks), but the clock signals may physically exist on-chip at the same time and therefore may still influence each other through crosstalk effects. Physically exclusive clocks, in contrast, cannot be		

*continued...*

	<p>physically present in the device at the same time (e.g., multiple clocks defined on the same clock pin). The Timing Analyzer does not currently analyze crosstalk explicitly. Instead, the timing models use extra guard bands to account for any potential crosstalk-induced delays. As a result, the Timing Analyzer currently treats asynchronous, logically_exclusive, and physically_exclusive clock groups the same. However, different parts of the Timing Analyzer may treat asynchronous and exclusive groups differently. Any commands that are affected by clock groups will say so in their help text. But, no distinction is made between logically and physically exclusive clock groups, since the only difference between them is how they affect crosstalk. The result of set_clock_groups is that all clocks in any group are cut from all clocks in every other group. The use of a single -group option tells the Timing Analyzer to cut this group of clocks from all other clocks in the design, including clocks that are created in the future. This command is similar to calling set_false_path from each clock in every group to each clock in every other group and vice versa, making set_clock_groups easier to specify for cutting clock domains. However, cutting clocks with set_clock_groups also affects the results of some other commands. Any commands that are affected by clock groups will say so in their help text.</p>						
<b>Example Usage</b>	<pre>project_open top create_timing_netlist create_clock -period 10.000 -name clkA [get_ports sysclk[0]] create_clock -period 10.000 -name clkB [get_ports sysclk[1]]  # Set clkA and clkB to be mutually exclusive clocks. set_clock_groups -logically_exclusive -group {clkA} -group {clkB}  # The previous line is equivalent to the following two commands. set_false_path -from [get_clocks clkA] -to [get_clocks clkB] set_false_path -from [get_clocks clkB] -to [get_clocks clkA]</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.27.21. set\_clock\_latency (::quartus::sdc)

The following table displays information for the set\_clock\_latency Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	set_clock_latency [-h   -help] [-long_help] [-clock <clock_list>] [-early] [-fall] [-late] [-rise] -source <delay> <targets>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-clock <clock_list>	Valid clock destinations (string patterns are matched using Tcl string matching)
	-early	Specifies the early clock latency
	-fall	Specifies the falling transition clock latency
	-late	Specifies the late clock latency
	-rise	Specifies the rising transition clock latency
	-source	Specifies the source clock latency
	<delay>	Latency delay value
<b>Description</b>	Specifies clock latency for a given clock or clock target. There are two types of latency: network and source. Network latency is the clock network delay between the clock and register clock pins. Source latency is the clock network delay between the clock and its source (e.g., the system clock or base clock of a generated clock). The Timing Analyzer automatically computes network latencies for all register and generated clocks. Overriding clock network latencies is not supported by the Timing Analyzer. Therefore, the -source option must always be specified. You can apply clock latency to a clock, which affects all targets of the clock, or to a specific clock target. If you specify a specific clock	

*continued...*

	<p>target that is driven by more than one clock, use the <code>-clock</code> option to specify which clock to use. Latencies assigned to a clock target override any latencies assigned to a clock. Different clock latencies can be specified for early (<code>-early</code>) and late (<code>-late</code>) latencies, as well as for rising edges (<code>-rise</code>) and falling edges (<code>-fall</code>). If only some combinations are specified, the other combinations are used by default. For example, if only a <code>-rise -early</code> latency and a <code>-fall -early</code> latency are specified, then the <code>-rise -late</code> latency is assumed to be the same as the <code>-rise -early</code> latency and the <code>-fall -late</code> latency is assumed to be the same as the <code>-fall -early</code> latency. If neither <code>-rise</code> nor <code>-fall</code> are used or neither <code>-early</code> nor <code>-late</code> are used, then the latency applies to both conditions. Source latency can also be assigned to generated clocks. This may be useful for specifying board level delays from a clock output port to a clock input port when the clock input port is acting as a feedback clock. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <code>use_timing_analyzer_styleEscaping</code> command for details.</p>									
<b>Example Usage</b>	<pre>create_clock -name SYSCLK -period 10.000 [get_ports inclk] create_generated_clock -name OUTCLK -divide_by 1 -source [get_ports inclk] [get_ports outclk] create_generated_clock -name FDBKCLK -divide_by 1 -source [get_ports outclk] [get_ports fdbkclk]  # Apply a simple 2.000 ns source latency to the system clock. set_clock_latency -source 2.000 [get_clocks SYSCLK]  # Specify feedback clock latencies between output port outclk # and the input port fdbkclk. set_clock_latency -source -late -rise 0.800 [get_clocks FDBKCLK] set_clock_latency -source -late -fall 0.750 [get_clocks FDBKCLK] set_clock_latency -source -early -rise 0.500 [get_clocks FDBKCLK] set_clock_latency -source -early -fall 0.460 [get_clocks FDBKCLK]</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.								

### 3.1.27.22. `set_clock_uncertainty` (::quartus::sdc)

The following table displays information for the `set_clock_uncertainty` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	<code>set_clock_uncertainty [-h   -help] [-long_help] [-add] [-enable_same_physical_edge] [-fall_from &lt;fall_from_clock&gt;] [-fall_to &lt;fall_to_clock&gt;] [-from &lt;from_clock&gt;] [-hold] [-rise_from &lt;rise_from_clock&gt;] [-rise_to &lt;rise_to_clock&gt;] [-setup] [-to &lt;to_clock&gt;] &lt;uncertainty&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-add</code>	Specifies that this assignment is an addition to the clock uncertainty derived by <code>derive_clock_uncertainty</code> call
	<code>-enable_same_physical_edge</code>	Enable setting uncertainty value for same physical clock edge
	<code>-fall_from &lt;fall_from_clock&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
	<code>-fall_to &lt;fall_to_clock&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
	<code>-from &lt;from_clock&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
	<code>-hold</code>	Only apply the uncertainty value to hold and removal checks

*continued...*

	-rise_from <rise_from_clock>	Valid destinations (string patterns are matched using Tcl string matching)	
	-rise_to <rise_to_clock>	Valid destinations (string patterns are matched using Tcl string matching)	
	-setup	Only apply the uncertainty value to setup and recovery checks	
	-to <to_clock>	Valid destinations (string patterns are matched using Tcl string matching)	
	<uncertainty>	Uncertainty	
<b>Description</b>	Specifies clock uncertainty or skew for clocks for clock-to-clock transfers. You can specify uncertainty separately for setup and hold, and you can specify separate rising and falling clock transitions. If you omit to specify -setup or -hold, the uncertainty value will be applied to both analysis types. Similarly, if you omit to specify rising or falling clock transitions, the uncertainty value will be applied to both transitions. The setup uncertainty is subtracted from the data required time for each applicable path, and the hold uncertainty is added to the data required time for each applicable path. Intel Quartus Prime software computes clock uncertainty for every clock transfer. For particular transfers, you can use the set_clock_uncertainty assignment to override the automatically derived value, or you can specify the -add option to add to the automatically derived value. Note that the -add option is only relative to the automatically derived value. If multiple set_clock_uncertainty assignments apply to the same clock transfer, the later value overrides the earlier ones, regardless of whether the -add option was used. Note: The Timing Analyzer does not apply clock uncertainty to transfers involving the same physical launch and latch edge (that is, the latch and launch edges are the same edge of a clock source and occur at the same time) by default. Such transfers typically occur in hold analysis, but may also occur in setup analysis with a multicycle value of 0. You can use the -enable_same_physical_edge option to override this behavior. The values for -from, -to, and similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.		
<b>Example Usage</b>	<pre>set_clock_uncertainty -setup -rise_from clk1 -fall_to clk2 200ps</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.27.23. set\_disable\_timing (::quartus::sdc)

The following table displays information for the set\_disable\_timing Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346		
<b>Syntax</b>	<pre>set_disable_timing [-h   -help] [-long_help] [-from &lt;name&gt;] [-to &lt;name&gt;] &lt;cells&gt;</pre>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-from <name>	Valid source pin suffix	
	-to <name>	Valid destination pin suffix	
	<cells>	List of cells	
<b>Description</b>	Disables a timing edge (arc) from inside a given cell or cells. Disabling a timing edge prevents timing analysis through that edge. If either -from or -to (or both) are unspecified, the missing value or values are replaced by a "*" character. The values of the -from and -to are valid pin suffixes. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.		
<b>Example Usage</b>	<pre>set_disable_timing -from datain -to combout A B set_disable_timing -from carryin *</pre>		
<i>continued...</i>			

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.27.24. set\_false\_path (::quartus::sdc)

The following table displays information for the set\_false\_path Tcl command:

Tcl Package and Version	Belongs to ::quartus::sdc on page 346	
Syntax	<pre>set_false_path [-h   -help] [-long_help] [-fall_from &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-hold] [-latency_insensitive] [-no_synchronizer] [-rise_from &lt;names&gt;] [-rise_to &lt;names&gt;] [-setup] [-through &lt;names&gt;] [-to &lt;names&gt;]</pre>	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-fall_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-hold	Specifies the false_path value (applies only to clock hold or removal checks)
	-latency_insensitive	Mark this false path as one that should still be optimized
	-no_synchronizer	Prevent this false path from triggering a synchronizer
	-rise_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-rise_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-setup	Specifies the false_path value (applies only to clock setup or recovery checks)
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
Description	<p>Specifies a false-path exception, removing (or cutting) paths from timing analysis. The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection only if the clock collection also generates the desired assignment. Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths. If the -latency_insensitive flag is set, the Fitter will be allowed to freely insert additional pipelining stages on any paths that are cut by the exception. These pipelined stages will be retimed to improve performance, but the timing requirements of the paths will still be ignored, just like for ordinary false paths. Without this flag, the Fitter will not perform any optimizations on the cut paths. If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin. The -through values are collections of pins or nets in the design. An</p>	

*continued...*

	<p>exception applied through a node in the design applies only to paths through the specified node. The Timing Analyzer allows you to specify the -through argument multiple times to describe paths that go through multiple points. For instance, users can select all paths that go through node X, and then go through node Y. This helps you narrow down and select the specific paths that you are interested in. The -rise_from and -fall_from options can be used in place of the -from destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node, taking into consideration any logical inversions along the clock path. The -from option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge. The -rise_to and -fall_to options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value, taking into consideration any logical inversions that are along the clock path. The -setup and -hold options allow the false path to only be applied to the corresponding setup/recovery or hold/removal checks. The default if neither value is specified is to apply the false path to both -setup and -hold. The values of the -from, -to, -through, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_style_escaping command for details. See help for the set_clock_groups command for information.</p>						
<b>Example Usage</b>	<pre># Set a false-path between two unrelated clocks # See also set_clock_groups set_false_path -from [get_clocks clkA] -to [get_clocks clkB]  # Set a false-path for a specific path set_false_path -from [get_pins regA clk] -to [get_pins regB aclr]  # Set a false-path from a node to a falling clock set_false_path -from [get_pins regA clk] -fall_to [get_clocks clkB]</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.27.25. set\_input\_delay (::quartus::sdc)

The following table displays information for the set\_input\_delay Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	set_input_delay [-h   -help] [-long_help] [-add_delay] [-blackbox] -clock <name> [-clock_fall] [-fall] [-max] [-min] [-reference_pin <name>] [-rise] [-source_latency_included] <delay> <targets>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-add_delay	Create additional delay constraint instead of overriding previous constraints
	-blackbox	Create an assignment for a partition boundary port causing it to be treated as a port
	-clock <name>	Clock name
	-clock_fall	Specifies that input delay is relative to the falling edge of the clock
	-fall	Specifies the falling input delay at the port
	-max	Applies value as maximum data arrival time
	-min	Applies value as minimum data arrival time
	-reference_pin <name>	Specifies a pin or port in the design to which the input delay is relative

*continued...*

	<p><b>-rise</b></p> <p><b>-source_latency_included</b></p> <p><b>&lt;delay&gt;</b></p> <p><b>&lt;targets&gt;</b></p>	<p>Specifies the rising input delay at the port</p> <p>Specifies that input delay includes added source latency</p> <p>Time value</p> <p>List of input port type objects</p>															
<b>Description</b>	<p>Specifies the data arrival times at the specified input ports relative the clock specified by the <b>-clock</b> option. The clock must refer to a clock name in the design. Input delays can be specified relative to the rising edge (default) or falling edge (<b>-clock_fall</b>) of the clock. Input delays can be specified relative to a pin or a port (<b>-reference_pin</b>) in the clock network. Clock arrival times to the reference pin or port are added to data arrival times. If no <b>-reference_pin</b> is specified, if the input delay is specified relative to a generated clock with a single target, the clock arrival times to the generated clock are added to the data arrival time. If the generated clock has multiple targets, the worst case arrival time to those targets will be used. Input delays can already include clock source latency. By default the clock source latency of the related clock is added to the input delay value, but when the <b>-source_latency_included</b> option is specified, the clock source latency is not added because it was factored into the input delay value. The maximum input delay (<b>-max</b>) is used for clock setup checks or recovery checks and the minimum input delay (<b>-min</b>) is used for clock hold checks or removal checks. If only <b>-min</b> or <b>-max</b> (or neither) is specified for a given port, the same value is used for both. Separate rising (<b>-rise</b>) and falling (<b>-fall</b>) arrival times at the port can be specified. If only one of <b>-rise</b> and <b>-fall</b> are specified for a given port, the same value is used for both. By default, <b>set_input_delay</b> removes any other input delays to the port except for those with the same <b>-clock</b>, <b>-clock_fall</b>, and <b>-reference_pin</b> combination. Multiple input delays relative to different clocks, clock edges, or reference pins can be specified using the <b>-add_delay</b> option. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <b>use_timing_analyzer_styleEscaping</b> command for details.</p>																
<b>Example Usage</b>	<pre># Simple input delay with the same value for min/max and rise/fall: # 1) set on ports with names of the form myin* set_input_delay -clock clk 1.5 [get_ports myin*] # 2) set on all input ports set_input_delay -clock clk 1.5 [all_inputs]  # Input delay with respect to the falling edge of clock set_input_delay -clock clk -clock_fall 1.5 [get_ports myin*]  # Input delays for different min/max and rise/fall combinations set_input_delay -clock clk -max -rise 1.4 [get_ports myin*] set_input_delay -clock clk -max -fall 1.5 [get_ports myin*] set_input_delay -clock clk -min -rise 0.7 [get_ports myin*] set_input_delay -clock clk -min -fall 0.8 [get_ports myin*]  # Adding multiple input delays with respect to more than one clock set_input_delay -clock clkA -min 1.2 [get_ports myin*] set_input_delay -clock clkA -max 1.8 [get_ports myin*] set_input_delay -clock clkA -clock_fall 1.6 [get_ports myin*] -add_delay set_input_delay -clock clkB -min 2.1 [get_ports myin*] -add_delay set_input_delay -clock clkB -max 2.5 [get_ports myin*] -add_delay  # Specifying an input delay relative to an external clock output port set_input_delay -clock clk -reference_pin [get_ports clkout] 0.8 [get_ports myin*]  # Specifying an input delay relative to the clock pin of a register set_input_delay -clock clk -reference_pin [get_pins regA clk] 0.8 [get_ports myin*]</pre>																
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Incorrect collection type. Expected a collection of type &lt;string&gt;.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Options -&lt;string&gt; and -&lt;string&gt; are mutually exclusive. Specify only one of the two options.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use <b>create_timing_netlist</b> to create a timing netlist.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Incorrect collection type. Expected a collection of type <string>.	TCL_ERROR	1	ERROR: Options -<string> and -<string> are mutually exclusive. Specify only one of the two options.	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <b>create_timing_netlist</b> to create a timing netlist.	
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>															
TCL_OK	0	INFO: Operation successful															
TCL_ERROR	1	ERROR: Incorrect collection type. Expected a collection of type <string>.															
TCL_ERROR	1	ERROR: Options -<string> and -<string> are mutually exclusive. Specify only one of the two options.															
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <b>create_timing_netlist</b> to create a timing netlist.															

### 3.1.27.26. set\_input\_transition (::quartus::sdc)

The following table displays information for the `set_input_transition` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc</a> on page 346		
<b>Syntax</b>	<code>set_input_transition [-h   -help] [-long_help] [-clock &lt;name&gt;] [-clock_fall] [-fall] [-max] [-min] [-rise] &lt;transition&gt; &lt;ports&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-clock <name>	Clock name	
	-clock_fall	Specifies that input delay is relative to the falling edge of the clock	
	-fall	Specifies the falling output delay at the port	
	-max	Applies value as maximum data required time	
	-min	Applies value as minimum data required time	
	-rise	Specifies the rising output delay at the port	
	<transition>	Time value	
	<ports>	Collection or list of input or bidir ports	
<b>Description</b>	This constraint does not affect calculations performed by the Timing Analyzer. It only affects PrimeTime analysis. If you set this constraint in the Timing Analyzer the constraint is written out to the SDC file when you call <code>write_sdc</code> .		
<b>Example Usage</b>	<code>set_input_transition 50 [all_inputs]</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Incorrect collection type. Expected a collection of type <string>.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.27.27. set\_max\_delay (::quartus::sdc)

The following table displays information for the `set_max_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc</a> on page 346		
<b>Syntax</b>	<code>set_max_delay [-h   -help] [-long_help] [-fall_from &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-rise_from &lt;names&gt;] [-rise_to &lt;names&gt;] [-through &lt;names&gt;] [-to &lt;names&gt;] &lt;value&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-fall_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)	

*continued...*

	-fall_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)	
	-rise_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)	
	-rise_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)	
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	<value>	Time Value	
<b>Description</b>	Specifies a maximum delay exception for a given path. The maximum delay is similar to changing the setup relationship (latching clock edge - launching clock edge), except that it can be applied to input or output ports without input or output delays assigned to them. Maximum delays are always relative to any clock network delays (if the source or destination is a register) or any input or output delays (if the source or destination is a port). Therefore, input delays and clock latencies are added to the data arrival times. Clock latencies also added to data required times and output delays are subtracted from data required times. The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart to the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection but only if the clock collection also generates the desired assignment. Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths. If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin. The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node. The Timing Analyzer allows you to specify the -through argument multiple times to describe paths that go through multiple points. For instance, users can select all paths that go through node X, and then go through node Y. This helps you narrow down and select the specific paths that you are interested in. The -rise_from and -fall_from options can be used in place of the -from destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The "-from" option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge. The -rise_to and -fall_to options behave similarly to the "from1" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path. The values of the -from, -to, -through, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_style_escaping command for details.		
<b>Example Usage</b>	<pre># Apply a 10ns max delay between two unrelated clocks set_max_delay -from [get_clocks clkA] -to [get_clocks clkB] 10.000  # Apply a 2ns max delay for an input port (TSU) set_max_delay -from [get_ports in[*]] -to [get_registers *] 2.000  # Apply a 2ns max delay for an output port (TCO) set_max_delay -from [get_registers *] -to [get_ports out[*]] 2.000  # Apply a 2ns max delay for an input port to an output port (TPD) set_max_delay -from [get_ports in[*]] -to [get_ports out[*]] 2.000  # Apply a 2ns max delay for an input port only to nodes driven by # the rising edge of clock CLK set_max_delay -from [get_ports in[*]] -rise_to [get_clocks CLK] 2.000</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.27.28. set\_max\_time\_borrow (::quartus::sdc)

The following table displays information for the `set_max_time_borrow` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc</a> on page 346		
<b>Syntax</b>	<code>set_max_time_borrow [-h   -help] [-long_help] [-exact] &lt;value&gt; &lt;targets&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-exact</code>	Forces the time borrowed to be the exact value provided (if physically possible)	
	<code>&lt;value&gt;</code>	Time Value	
	<code>&lt;targets&gt;</code>	Collection or list of latches	
<b>Description</b>	Specifies the maximum borrowed time for level-sensitive latches. The actual borrowed time will be determined automatically, but will never exceed the amount you specify. For any latches without a <code>set_max_time_borrow</code> constraint, no limit will apply (except for the physical limit of what is possible on the device, as described below). By using the <code>-exact</code> option, you can bypass the automatic algorithm and specify the exact amount of borrowing at a given latch. For optimum results, using the automatic algorithm is recommended (ideally, without any <code>set_max_time_borrow</code> constraints). Time borrowing is specified with respect to the earliest possible time a signal can be clocked into the latch node. For example, for a positive latch, if the earliest possible arrival time of the rising clock edge is 1.025ns, then a signal that has an arrival time of 1.035ns (where this arrival time already includes the micro-setup time of the latch) will require at least 0.010ns of time borrowing. Regardless of how the borrowed time is determined (automatically without a limit, automatically with a <code>set_max_time_borrow</code> constraint, or manually with a <code>set_max_time_borrow -exact</code> constraint), the borrowed time can never exceed what is physically possible to borrow on the device. The maximum amount that can be borrowed is the period of time when the latch is open (e.g. half the clock period if the clock has a 50% duty cycle), but this time is reduced by clock propagation time spread and clock uncertainty between the latch-opening and latch-closing clock edges, and is further reduced by the closing-edge setup time of the latch. Some of these factors vary from corner to corner, as well as from clock to clock (if multiple clocks drive the latch). Time borrowing analysis will only occur in the Timing Analysis (Signoff) stage, or when manually running the Timing Analyzer. The Fitter will not utilize time borrowing information and will assume zero time borrowed. Thus, the use of level-sensitive latches with high-speed clocks is not recommended, unless other constraints (such as <code>set_max_delay</code> ) are manually set to ensure optimal Fitter behavior. The targets of this command must be level-sensitive latches (all other targets will be ignored). The targets can be specified as either a collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <code>use_timing_analyzer_styleEscaping</code> command for details.		
<b>Example Usage</b>	<pre># Borrow at most 3ns at all "lat*" latches: set_max_time_borrow 3 [get_registers lat*]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.

### 3.1.27.29. set\_min\_delay (::quartus::sdc)

The following table displays information for the `set_min_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	<pre>set_min_delay [-h   -help] [-long_help] [-fall_from &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-rise_from &lt;names&gt;] [-rise_to &lt;names&gt;] [-through &lt;names&gt;] [-to &lt;names&gt;] &lt;value&gt;</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-fall_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-rise_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-rise_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	<value>	Time Value
<b>Description</b>	<p>Specifies a minimum delay exception for a given path. The minimum delay is similar to changing the hold relationship (launching clock edge - latching clock edge), except that it can be applied to input or output ports without input or output delays assigned to them. Minimum delays are always relative to any clock network delays (if the source or destination is register) or any input or output delays (if the source or destination is a port). Therefore, input delays and clock latencies are added to the data arrival times. Clock latencies also added to data required times and output delays are subtracted from data required times. The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection, but only if the clock collection also generates the desired assignment. Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Also, applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths. If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin. The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node. The Timing Analyzer allows you to specify the -through argument multiple times to describe paths that go through multiple points. For instance, users can select all paths that go through node X, and then go through node Y. This helps you narrow down and select the specific paths that you are interested in. The -rise_from and -fall_from options can be used in place of the destination nodes specified using the -from option. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The -from option is the combination of both rising and falling "from" nodes. If the -from collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge. The -rise_to and -fall_to options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path. The values of the -from, -to, -through, and other similar options are either collections or a Tcl</p>	

*continued...*

	<p>list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <code>use_timing_analyzer_styleEscaping</code> command for details.</p>						
<b>Example Usage</b>	<pre># Apply a 1ns min delay between two unrelated clocks set_min_delay -from [get_clocks clkA] -to [get_clocks clkB] 0.000  # Apply a 1ns min delay for an input port (TH) set_min_delay -from [get_ports in[*]] -to [get_registers *] -.000  # Apply a 0.5ns min delay for an output port (MIN_TCO) set_min_delay -from [get_registers *] -to [get_ports out[*]] 0.500  # Apply a 0.5ns min delay for an input port to an output port (MIN_TPD) set_min_delay -from [get_ports in[*]] -to [get_ports out[*]] 0.500  # Apply a 0.5ns min delay for an input port only to nodes driven by # the falling edge of clock CLK set_max_delay -from [get_ports in[*]] -fall_to [get_clocks CLK] 0.500</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.27.30. `set_multicycle_path` (::quartus::sdc)

The following table displays information for the `set_multicycle_path` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc on page 346	
<b>Syntax</b>	<code>set_multicycle_path [-h   -help] [-long_help] [-end] [-fall_from &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-hold] [-rise_from &lt;names&gt;] [-rise_to &lt;names&gt;] [-setup] [-start] [-through &lt;names&gt;] [-to &lt;names&gt;] &lt;value&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-end	Specifies that the multicycle is relative to the destination clock waveform (default)
	-fall_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-hold	Specifies that the multicycle value applies to clock hold or removal checks
	-rise_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-rise_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-setup	Specifies that the multicycle value applies to clock setup or recovery checks (default)
	-start	Specifies that the multicycle is relative to the source clock waveform
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)

*continued...*

	<code>-to &lt;names&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)	
	<code>&lt;value&gt;</code>	Number of clock cycles	
<b>Description</b>	Specifies a multicycle exception for a given set of paths. Multicycles can be specified relative to the source clock (-start) or destination clock (-end). This is useful when the source clock and destination clock are operating at different frequencies. For example, if the source clock is twice as fast (half period) as the destination clock, a -start multicycle of 2 is usually required. Hold multicycles (-hold) are computed relative to setup multicycles (-setup). The value of the hold multicycle represents the number clock edges away from the default hold multicycle. The default hold multicycle value is 0. The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection but only if the clock collection also generates the desired assignment. Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Also, applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths. If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin. The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node. The Timing Analyzer allows you to specify the -through argument multiple times to describe paths that go through multiple points. For instance, users can select all paths that go through node X, and then go through node Y. This helps you narrow down and select the specific paths that you are interested in. The -rise_from and -fall_from options can be used in place of the "-from" destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The "from" option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge. The -rise_to and -fall_to options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path. The values of the -from, -to, -through, and similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_style_escaping command for details.		
<b>Example Usage</b>	<pre>create_clock -period 10.000 -name CLK [get_ports clk] create_generated_clock -divide_by 2 -source [get_ports clk] -name CLKDIV2 [get_registers clkdiv]  # Apply a source multicycle of 2 with a hold multicycle of 1 for all # paths from the CLK domain to the CLKDIV2 domain. set_multicycle_path -start -setup -from [get_clocks CLK] -to [get_clocks CLKDIV2] 2 set_multicycle_path -start -hold -from [get_clocks CLK] -to [get_clocks CLKDIV2] 1  # Apply a multicycle constraint of 3 (with a default hold multicycle of 0) for a # specific path in the design. set_multicycle_path -end -setup -from [get_pins rega clk] -to [get_pins regb *] 3  # Apply a multicycle constraint of 2 to a given cell, except for the reset pin. set_multicycle_path -end -setup -to [get_cells regb] 2 set_multicycle_path -end -setup -to [get_pins regb aclr] 1  #Apply a multicycle constraint of 3 rising from a clock and falling to a node set_multicycle_path -end -setup -rise_from [get_clocks CLK] -fall_to [get_pins datab] 3</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.

### 3.1.27.31. set\_output\_delay (::quartus::sdc)

The following table displays information for the `set_output_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::sdc</code> on page 346	
<b>Syntax</b>	<code>set_output_delay [-h   -help] [-long_help] [-add_delay] [-blackbox] -clock &lt;name&gt; [-clock_fall] [-fall] [-max] [-min] [-reference_pin &lt;name&gt;] [-rise] [-source_latency_included] &lt;delay&gt; &lt;targets&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-add_delay</code>	Create additional delay constraint instead of overriding previous constraints
	<code>-blackbox</code>	Create an assignment for a partition boundary port causing it to be treated as a port
	<code>-clock &lt;name&gt;</code>	Clock name
	<code>-clock_fall</code>	Specifies output delay relative to the falling edge of the clock
	<code>-fall</code>	Specifies the falling output delay at the port
	<code>-max</code>	Applies value as maximum data required time
	<code>-min</code>	Applies value as minimum data required time
	<code>-reference_pin &lt;name&gt;</code>	Specifies a pin or port in the design to which the output delay is relative
	<code>-rise</code>	Specifies the rising output delay at the port
	<code>-source_latency_included</code>	Specifies input delay already includes added source latency
<b>Description</b>	<code>&lt;delay&gt;</code>	Time value
	<code>&lt;targets&gt;</code>	Collection or list of output ports
<b>Example Usage</b>	<pre># Simple output delay with the same value for min/max and rise/fall: # 1) set on ports with names of the form myout* set_output_delay -clock clk 0.5 [get_ports myout*] # 2) set on all output ports set_output_delay -clock clk 0.5 [all_outputs]  # Output delay with respect to the falling edge of clock</pre>	

*continued...*

```

set_output_delay -clock clk -clock_fall 0.5 [get_ports myout*]

# Output delays for different min/max and rise/fall combinations
set_output_delay -clock clk -max -rise 0.5 [get_ports myout*]
set_output_delay -clock clk -max -fall 0.4 [get_ports myout*]
set_output_delay -clock clk -min -rise 0.4 [get_ports myout*]
set_output_delay -clock clk -min -fall 0.3 [get_ports myout*]

# Adding multiple output delays with respect to more than one clock
set_output_delay -clock clkA -min 0.2 [get_ports myout*]
set_output_delay -clock clkA -max 0.8 [get_ports myout*]
set_output_delay -clock clkA -clock_fall 0.6 [get_ports myout*] -add_delay
set_output_delay -clock clkB -min 1.1 [get_ports myout*] -add_delay
set_output_delay -clock clkB -max 1.5 [get_ports myout*] -add_delay

# Specifying an output delay relative to an external clock output port
set_output_delay -clock clk -reference_pin [get_ports clkout] 0.8 [get_ports myout*]

# Specifying an output delay relative to the clock pin of a register
set_output_delay -clock clk -reference_pin [get_pins regA|clk] 0.8 [get_ports myout*]

```

Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Incorrect collection type. Expected a collection of type <string>.
TCL_ERROR	1		ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28. ::quartus::sdc\_ext

The following table displays information for the **::quartus::sdc\_ext** Tcl package:

<b>Tcl Package and Version</b>	::quartus::sdc_ext 2.0
<b>Description</b>	Timing Constraints not defined in the SDC Spec Version 1.5 are implemented in this package. Any command in this package can be specified in a Timing Analyzer SDC file.
<b>Availability</b>	<p>This package is loaded by default in the following executable:</p> <pre>quartus_sta</pre> <p>This package is available for loading in the following executable:</p> <pre>quartus_fit</pre>
<b>Tcl Commands</b>	<pre> derive_clock_uncertainty (::quartus::sdc_ext) on page 376 derive_pll_clocks (::quartus::sdc_ext) on page 376 disable_min_pulse_width (::quartus::sdc_ext) on page 377 get_active_clocks (::quartus::sdc_ext) on page 378 get_fanins (::quartus::sdc_ext) on page 379 get_fanouts (::quartus::sdc_ext) on page 380 get_keepers (::quartus::sdc_ext) on page 381 get_nodes (::quartus::sdc_ext) on page 381 get_partitions (::quartus::sdc_ext) on page 382 get_registers (::quartus::sdc_ext) on page 383 remove_annotated_delay (::quartus::sdc_ext) on page 384 remove_clock (::quartus::sdc_ext) on page 384 reset_timing_derate (::quartus::sdc_ext) on page 385 set_active_clocks (::quartus::sdc_ext) on page 385 set_annotated_delay (::quartus::sdc_ext) on page 386 set_data_delay (::quartus::sdc_ext) on page 387 set_max_skew (::quartus::sdc_ext) on page 389 set_net_delay (::quartus::sdc_ext) on page 390 set_scc_mode (::quartus::sdc_ext) on page 391 set_time_format (::quartus::sdc_ext) on page 392 set_timing_derate (::quartus::sdc_ext) on page 392 </pre>

### 3.1.28.1. derive\_clock\_uncertainty (::quartus::sdc\_ext)

The following table displays information for the derive\_clock\_uncertainty Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	derive_clock_uncertainty [-h   -help] [-long_help] [-add] [-overwrite]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-add	Adds results to user-defined clock uncertainty assignments	
	-overwrite	Overwrites user-defined clock uncertainty assignments	
<b>Description</b>	Applies inter-clock, intra-clock and I/O interface uncertainties based on timing model characterization. This command calculates and applies setup and hold clock uncertainties for each clock-to-clock transfer found in the design. The calculation of the uncertainties is delayed until the next update_timing_netlist call. To get I/O interface uncertainty in addition to inter-clock and intra-clock uncertainties, create a virtual clock to represent an off-chip clock for input or output delay specification and assign delays to input/output ports with set_input_delay and set_output_delay commands that specify the virtual clock. If set_input_delay and set_output_delay commands specifying a non- virtual clock are called, derive_clock_uncertainty applies either inter-clock or intra-clock uncertainty for that clock transfer since those transfers represent a clock-to-clock domain rather than an I/O-to-register clock domain. The set_clock_uncertainty calls will override the derived values for a source clock and destination clock pair unless either the set_clock_uncertainty command or the derive_clock_uncertainty command specified the -add option, in which case the values are added. Specifying the -overwrite option will instead cause all set_clock_uncertainty commands to be ignored. Previous set_clock_uncertainty assignments can also be manually removed by using the remove_clock_uncertainty command. Note that this command is called automatically and the user only needs to manually call it to specify the -add or -overwrite options.		
<b>Example Usage</b>	<pre># create a virtual clock create_clock -name virtual -period 1  # apply input/output delays with the virtual clock to get # I/O interface uncertainties set_input_delay -clock virtual -add_delay 0 [all_inputs] set_output_delay -clock virtual -add_delay 0 [all_outputs]  # call derive_clock_uncertainty. results will be calculated # at the next update_timing_netlist call derive_clock_uncertainty  update_timing_netlist</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.2. derive\_pll\_clocks (::quartus::sdc\_ext)

The following table displays information for the derive\_pll\_clocks Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	derive_pll_clocks [-h   -help] [-long_help] [-create_base_clocks] [-use_net_name]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-create_base_clocks	Creates base clocks on input clock ports of the design that are feeding the PLL	
<i>continued...</i>			

	-use_net_name	Use net names as clock names	
<b>Description</b>	NOTE: This command is no longer supported for Stratix 10 and later families. Identifies PLLs or similar resources in the design and creates generated clocks for their output clock pins. Multiple generated clocks may be created for each output clock pin if the PLL is using clock switchover, one for the inclk[0] input clock pin and one for the inclk[1] input clock pin. By default this command does not create base clocks on input clock ports that are driving the PLL. When you use the create_base_clocks option, derive_pll_clocks also creates the base clock on an input clock port deriving the PLL. This option does not overwrite an existing clock. By default the clock name is the same as the output clock pin name. To use the net name, use the -use_net_name option. Note that this command is not supported for Stratix 10 and later device families. The only families that still have support for this command are Arria 10 and Cyclone 10GX. The reason why this command has been deprecated is because all PLL clocks are now automatically generated by the SDC files generated alongside the PLL IP. No user action is required.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist  # Create the base clock for the input clock port driving the PLL create_clock -period 10.0 [get_ports sysclk]  # Create the generated clocks for the PLL. derive_pll_clocks  update_timing_netlist  # Other user actions report_timing  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.3. disable\_min\_pulse\_width (::quartus::sdc\_ext)

The following table displays information for the disable\_min\_pulse\_width Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	disable_min_pulse_width [-h   -help] [-long_help] <targets>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	<targets>		Elements to disable MPW checks for
<b>Description</b>	Allows you to disable minimum pulse width checks for specified targets. If the target is a clock collection, all MPW checks along that clock path will be disabled. Otherwise, MPW checks for the elements in the passed target collection will be disabled.		
<b>Example Usage</b>	<pre>disable_min_pulse_width -targets reg[*] disable_min_pulse_width -targets [get_clocks {clkA}]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.4. `get_active_clocks` (::quartus::sdc\_ext)

The following table displays information for the `get_active_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	<code>get_active_clocks [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Returns the collection of active clocks for timing analysis. The active clocks are the clocks specified in the most recent call to <code>set_active_clocks</code> , or all clocks if <code>set_active_clocks</code> has not been called.		
<b>Example Usage</b>	<pre># Set some active clocks set_active_clocks [get_clocks {clk1 clk2 sysclk*}]  # Update the active clocks to exclude clk1 set_active_clocks [remove_from_collection [get_active_clocks] [get_clocks clk1]]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.28.5. `get_assignment_groups` (::quartus::sdc\_ext)

The following table displays information for the `get_assignment_groups` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 0		
<b>Syntax</b>	<code>get_assignment_groups [-h   -help] [-long_help] [-keepers] [-ports] [-registers] &lt;name&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-keepers</code>	Returns a keeper collection from the assignment group matching the <name>	
	<code>-ports</code>	Returns a port collection from the assignment group matching the <name>	
	<code>-registers</code>	Returns a register collection from the assignment group matching the <name>	
	<code>&lt;name&gt;</code>	Assignment group name	
<b>Description</b>	Returns a collection of <keepers> <registers> <ports> for the assignment group that matches <name>. This command can be used to retrieve the assignment group created and saved in the Quartus Prime Settings File. The options -keepers, -registers and -ports are mutually exclusive. If no option is specified, the keeper collection is returned by default.		
<b>Example Usage</b>	<code>get_assignment_groups my_assignments -registers</code>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.6. get\_fanins (::quartus::sdc\_ext)

The following table displays information for the `get_fanins` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375		
<b>Syntax</b>	<code>get_fanins [-h   -help] [-long_help] [-asynch] [-clock] [-inverting_paths] [-no_logic] [-non_inverting_paths] [-stop_at_clocks] [-synch] [-through &lt;names&gt;] &lt;filter&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-asynch</code>	Traverse through asynch edges	
	<code>-clock</code>	Traverse through clock edges	
	<code>-inverting_paths</code>	Only follow inverting combinational paths	
	<code>-no_logic</code>	Do not follow combinational paths	
	<code>-non_inverting_paths</code>	Only follow non-inverting combinational paths	
	<code>-stop_at_clocks</code>	Return clock targets as fanin/fanouts rather than traversing through them	
	<code>-synch</code>	Traverse through synch edges	
	<code>-through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)	
<b>Description</b>	<filter>		
	Returns a collection of fanin ports, registers (and optionally clock targets) reachable from the <filter> in the design. When you supply the <code>-no_logic</code> option, <code>get_fanins</code> ignores paths that pass through combinational logic elements other than buffers and inverters. NOTE: the <code>-no_logic</code> option does not consider logic absorbed into the cells of the <filter> nor the cells of fanin registers, ports or clock targets. When you use <code>-synch</code> , <code>-asynch</code> , or <code>-clock</code> options, <code>get_fanins</code> traverses the netlist through corresponding edges. You can specify more than one of these options. If you do not specify any of these three options, the command does not ignore any paths. When the <code>-non_inverting_paths</code> option is used, <code>no_logic</code> does not follow any paths that includes odd number of inverters. Similarly, when the <code>-inverting_paths</code> option is used, <code>no_logic</code> does not follow any paths that includes even number of inverters. Both the <code>-non_inverting_paths</code> and <code>-inverting_paths</code> options require the <code>-no_logic</code> option and are mutually exclusive. When the <code>-through</code> option is used, only the fanins that can be reached by going through those nodes are returned. When <code>-stop_at_clocks</code> is used, combinational clock targets may be returned (in addition to clock or non-clock registers and ports), and registers or ports that can only be reached by traversing through a clock target will not be returned. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for the <code>use_timing_analyzer_styleEscaping</code> command for details.		
<b>Example Usage</b>	<pre>set fanins [get_fanins \$item -synch -clock] foreach_in_collection fanin_keeper \$fanins {     lappend fanin_keeper_list [get_node_info \$fanin_keeper -name] }  set fanins_no_logic [get_fanins \$item -no_logic -asynch] foreach_in_collection fanin_keeper \$fanins_no_logic {     lappend fanin_keeper_list_no_logic [get_node_info \$fanin_keeper -name] }  #-through example get_fanins inst18 -through inst11</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.7. `get_fanouts` (::quartus::sdc\_ext)

The following table displays information for the `get_fanouts` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375	
<b>Syntax</b>	<code>get_fanouts [-h   -help] [-long_help] [-asynch] [-clock] [-inverting_paths] [-no_logic] [-non_inverting_paths] [-stop_at_clocks] [-synch] [-through &lt;names&gt;] &lt;filter&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-asynch</code>	Traverse through asynchronous edges
	<code>-clock</code>	Traverse through clock edges
	<code>-inverting_paths</code>	Only follow inverting combinational paths
	<code>-no_logic</code>	Do not follow combinational paths
	<code>-non_inverting_paths</code>	Only follow non-inverting combinational paths
	<code>-stop_at_clocks</code>	Return clock targets as fanin/fanouts rather than traversing through them
	<code>-synch</code>	Traverse through synchronous edges
	<code>-through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)
<b>Description</b>	<code>&lt;filter&gt;</code>	
	Valid starting nodes (string patterns are matched using Tcl string matching or collection)	
<b>Example Usage</b>	<pre>set fanouts [get_fanouts \$item] foreach_in_collection fanout_keeper \$fanouts {     lappend fanout_keeper_list [get_node_info \$fanout_keeper -name] }  set fanouts_no_logic [get_fanouts \$item -no_logic] foreach_in_collection fanout_keeper \$fanouts_no_logic {     lappend fanout_keeper_list_no_logic [get_node_info \$fanout_keeper -name] }</pre>	

*continued...*

	<pre># Using through option to find the fanout registers whose enable input is # connected to the signal while ignoring the inverting paths. get_fanouts inst1 -no_logic -non_inverting_paths -through [get_pins -hierarchical * ena]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.8. get\_keepers (::quartus::sdc\_ext)

The following table displays information for the get\_keepers Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	get_keepers [-h   -help] [-long_help] [-no_duplicates] [-nocase] [-nowarn] [<filter> ]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-no_duplicates	Do not match duplicated keeper names	
	-nocase	Specifies the matching of node names to be case-insensitive	
	-nowarn	Do not issue warning messages about unmatched patterns	
	<filter>	Valid destinations (string patterns are matched using Tcl string matching)	
<b>Description</b>	Returns a collection of non-combinational or "keeper" nodes in the design. The default matching scheme returns not only non-combinational nodes whose names match the specified filter, but also non-combinational nodes duplicated from these keepers (refers to cells are automatically generated by Quartus from these keepers). Use the -no_duplicates option to exclude duplicated keepers. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set kprs [get_keepers *reg*] foreach_in_collection kpr \$kprs {     puts [get_object_info -name \$kpr] }  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.9. get\_nodes (::quartus::sdc\_ext)

The following table displays information for the get\_nodes Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	get_nodes [-h   -help] [-long_help] [-no_duplicates] [-nocase] [-nowarn] [<filter> ]		
<b>Arguments</b>	-h   -help	Short help	

*continued...*

	<table border="1"> <tr> <td>-long_help</td><td>Long help with examples and possible return values</td></tr> <tr> <td>-no_duplicates</td><td>Do not match duplicated node names</td></tr> <tr> <td>-nocase</td><td>Specifies the matching of node names to be case-insensitive</td></tr> <tr> <td>-nowarn</td><td>Do not issue warning messages about unmatched patterns</td></tr> <tr> <td>&lt;filter&gt;</td><td>Valid destinations (string patterns are matched using Tcl string matching)</td></tr> </table>	-long_help	Long help with examples and possible return values	-no_duplicates	Do not match duplicated node names	-nocase	Specifies the matching of node names to be case-insensitive	-nowarn	Do not issue warning messages about unmatched patterns	<filter>	Valid destinations (string patterns are matched using Tcl string matching)
-long_help	Long help with examples and possible return values										
-no_duplicates	Do not match duplicated node names										
-nocase	Specifies the matching of node names to be case-insensitive										
-nowarn	Do not issue warning messages about unmatched patterns										
<filter>	Valid destinations (string patterns are matched using Tcl string matching)										
<b>Description</b>	Returns a collection of nodes in the design. The default matching scheme returns not only nodes whose names match the specified filter, but also nodes duplicated from these nodes (refers to cells are automatically generated by Quartus from these nodes). Use the -no_duplicates option to not include duplicated nodes. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use_timing_analyzer_styleEscaping command for details.										
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set nodes [get_nodes *name*] foreach_in_collection node \$nodes {     puts [get_object_info -name \$node] }  delete_timing_netlist project_close</pre>										
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th> <th><b>Code</b></th> <th><b>String Return</b></th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>									
TCL_OK	0	INFO: Operation successful									
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.									

### 3.1.28.10. get\_partitions (::quartus::sdc\_ext)

The following table displays information for the get\_partitions Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375	
<b>Syntax</b>	get_partitions [-h   -help] [-long_help] [-cell] [-hierarchical] [-nocase] [<filter>]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-cell	Returns a cell collection inside the partitions matching the <filter>
	-hierarchical	Specifies if hierarchical searching method should be used
	-nocase	Specifies the matching of node names to be case-insensitive
	<filter>	Valid partitions (string patterns are matched using Tcl string matching)
<b>Description</b>	Returns a collection of partitions matching the filter by default. All partition names in the collection match the specified pattern. Wildcards can be used to select multiple partitions at once. The -cell option creates and returns the collection of cells found inside the partitions matching the <filter> instead of returning a partition collection. There are three Tcl string matching schemes available with this command: default, -hierarchical, and -no_case. When using the default matching scheme, pipe characters separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. The default matching scheme does not force the search to proceed recursively down the hierarchy. Using the hierarchical matching scheme forces the search to proceed recursively down the hierarchy. The -nocase matching scheme	

*continued...*

	uses case-insensitive matching behavior. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.									
<b>Example Usage</b>	<pre>#Get the partitions matching the filter get_partitions *</pre> <pre>#Get the collection of cells inside partitions matching the filter get_partitions * -cell</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: This command is not supported in this version of the software.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: This command is not supported in this version of the software.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: This command is not supported in this version of the software.								

### 3.1.28.11. `get_registers (::quartus::sdc_ext)`

The following table displays information for the `get_registers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375				
<b>Syntax</b>	<code>get_registers [-h   -help] [-long_help] [-latches] [-no_duplicates] [-nocase] [-nowarn] [ &lt;filter&gt; ]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-latches</code>	Get only latches that match the filter			
	<code>-no_duplicates</code>	Do not match duplicated register names			
	<code>-nocase</code>	Specifies the matching of node names to be case-insensitive			
	<code>-nowarn</code>	Do not issue warning messages about unmatched patterns			
	<code>&lt;filter&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)			
<b>Description</b>	Returns a collection of registers in the design. The default matching scheme returns not only registers whose names match the specified filter, but also returns registers duplicated from these registers (cells automatically generated from these registers by the Quartus Prime software). Use the <code>-no_duplicates</code> option to exclude duplicated registers. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the <code>use_timing_analyzer_styleEscaping</code> command for details.				
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set regs [get_registers *reg*] foreach_in_collection reg \$regs {     puts [get_object_info -name \$reg] }  delete_timing_netlist project_close</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.		

### 3.1.28.12. remove\_annotated\_delay (::quartus::sdc\_ext)

The following table displays information for the `remove_annotated_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375		
<b>Syntax</b>	<code>remove_annotated_delay [-h   -help] [-long_help] -all</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-all	Specifies removal of all annotated delays	
<b>Description</b>	Removes annotated delays from the design.		
<b>Example Usage</b>	<pre># annotate delay set_annotated_delay -net -from [get_pins clk] 0.1 update_timing_netlist  # remove all annotated delays remove_annotated_delay -all update_timing_netlist</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.13. remove\_clock (::quartus::sdc\_ext)

The following table displays information for the `remove_clock` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375		
<b>Syntax</b>	<code>remove_clock [-h   -help] [-long_help] [-all] [ &lt;clock_list&gt; ]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-all	Removes all clocks from the design	
	<clock_list>	Clock(s) to be removed	
<b>Description</b>	Removes the specified clock(s) from the design.		
<b>Example Usage</b>	<pre># Create a clock and then remove it. create_clock -period 10 -name CLK [get_ports clk] remove_clock CLK</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.14. reset\_timing\_derate (::quartus::sdc\_ext)

The following table displays information for the `reset_timing_derate` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	<code>reset_timing_derate [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Resets all derate factors set on the design.		
<b>Example Usage</b>	<pre># set timing derate set_timing_derate -late 0.2 [get_cells *] update_timing_netlist  # reset all derate factors reset_timing_derate update_timing_netlist</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.15. set\_active\_clocks (::quartus::sdc\_ext)

The following table displays information for the `set_active_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	<code>set_active_clocks [-h   -help] [-long_help] &lt;clocks&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>&lt;clocks&gt;</code>	List or collection of clocks	
<b>Description</b>	Sets the list of active clocks for timing analysis. All other clocks not in the list or collection are considered inactive. Timing analysis is only performed on active clocks. All clocks are active by default. Generated clocks that are generated from inactive clocks are considered inactive. Therefore, to make a generated clock active, specify both the parent and generated clock when calling <code>set_active_clocks</code> . To reset all clocks to active, call "set_active_clocks *" or "set_active_clocks [all_clocks]". The <code>set_active_clocks</code> command does not affect all reports. For example, inactive clocks are still reported by <code>report_clocks</code> , <code>report_clock_transfers</code> , and similar commands.		
<b>Example Usage</b>	<pre># Only analyze clk1 set_active_clocks [get_clocks clk1]  # Only analyze clk2 set_active_clocks [get_clocks clk2]  # Analyze all clocks set_active_clocks [all_clocks]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.16. set\_annotated\_delay (::quartus::sdc\_ext)

The following table displays information for the `set_annotated_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375	
<b>Syntax</b>	<code>set_annotated_delay [-h   -help] [-long_help] [-cell] [-ff] [-fr] [-from &lt;names&gt;] [-max] [-min] [-net] [-operating_conditions &lt;operating_conditions&gt;] [-rf] [-rr] [-to &lt;names&gt;] &lt;delay&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-cell</code>	Specifies that cell delay must be set
	<code>-ff</code>	Specifies that FF delay must be set
	<code>-fr</code>	Specifies that FR delay must be set
	<code>-from &lt;names&gt;</code>	Valid source pins or ports (string patterns are matched using Tcl string matching)
	<code>-max</code>	Specifies that only max delay should be set
	<code>-min</code>	Specifies that only min delay should be set
	<code>-net</code>	Specifies that net delay must be set
	<code>-operating_conditions &lt;operating_conditions&gt;</code>	Operating conditions Tcl object
	<code>-rf</code>	Specifies that RF delay must be set
	<code>-rr</code>	Specifies that RR delay must be set
<b>Description</b>	Valid destination pins or ports (string patterns are matched using Tcl string matching)	
	<delay>	
The delay value in default time units		
<b>Example Usage</b>	<pre>set_annotated_delay -cell 100 -from A B C datain -to A B C combout -rr -ff set_annotated_delay -net 100 -to A carryin update_timing_netlist  # To clear all net delays set_annotated_delay -net 0 update_timing_netlist  # To remove all annotated delay assignments remove_annotated_delay -all update_timing_netlist</pre>	

*continued...*

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.28.17. set\_data\_delay (::quartus::sdc\_ext)

The following table displays information for the set\_data\_delay Tcl command:

Tcl Package and Version	Belongs to ::quartus::sdc_ext on page 375	
Syntax	<pre>set_data_delay [-h   -help] [-long_help] [-add_latch_clock] [-add_launch_clock] [-allow_destination_borrowing] [-fall_from &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-get_value_from_clock_period &lt;src_clock_period dst_clock_period min_clock_period max_clock_period&gt;] [-override] [-rise_from &lt;names&gt;] [-rise_to &lt;names&gt;] [-through &lt;names&gt;] [-to &lt;names&gt;] [-value_multiplier &lt;multiplier&gt;] [ &lt;value&gt; ]</pre>	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-add_latch_clock	Include the latch clock path in timing analysis
	-add_launch_clock	Include the launch clock path in timing analysis
	-allow_destination_borrowing	Allow time borrowing at the destination
	-fall_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-get_value_from_clock_period <src_clock_period dst_clock_period min_clock_period max_clock_period>	Compute constraint as a multiple of the clock period
	-override	Make this constraint override non-datapath-only constraints, instead of applying it in addition to them
	-rise_from <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-rise_to <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-value_multiplier <multiplier>	Value by which the clock period should be multiplied to compute requirement
	<value>	Time Value
Description	<p>Specifies a maximum datapath delay exception for a given path. NOTE: This constraint should only be used for clock domain crossing paths. See the end of this help section for more information. The maximum delay analysis includes Tco of the launching register, and Tsu of the latching register. By default, it does not include clock arrival times at the launching or latching register. To include launch clock arrival times, use the -allow_launch_clock option. To include latch clock arrival times, use the -allow_latch_clock option.</p>	

*continued...*

allow\_latch\_clock option. If the path starts or ends at a port, the analysis does not include delays due to set\_input\_delay or set\_output\_delay. Use -get\_value\_from\_clock\_period to set the delay requirement for each path to be equal to the launching or latching clock period, or whichever of the two has a smaller or larger period. If -value\_multiplier is used, the requirement will be multiplied by that value. If there are no clocks clocking the endpoints of the path (e.g. if the path begins or ends at an unconstrained I/O), the constraint will be ignored. The datapath delay constraint is applied in addition to other constraints on the given path, including the default constraint. Furthermore, the datapath delay constraint is analyzed independently from other SDC constraints, including set\_false\_path and set\_clock\_groups, and cannot be overridden by other SDC constraints. For example, you can use set\_data\_delay to specify an upper limit on logic and routing delay for paths cut by set\_false\_path. The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin. The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node. The Timing Analyzer allows you to specify the -through argument multiple times to describe paths that go through multiple points. For instance, users can select all paths that go through node X, and then go through node Y. This helps you narrow down and select the specific paths that you are interested in. The -rise\_from and -fall\_from options can be used in place of the -from destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The "-from" option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge. The -rise\_to and -fall\_to options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path. The values of the -from, -to, -through, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See help for the use\_timing\_analyzer\_style\_escaping command for details. If the source of a path with a set\_data\_delay constraint has any time borrowed, the delay budget will be reduced by the time borrowed. By default, the delay budget will not be increased by time borrowed at the destination of a path constrained by a set\_data\_delay constraint, and negative slack on a set\_data\_delay constraint will not cause time borrowing to happen. To change this behavior, use the -allow\_destination\_borrowing option. set\_data\_delay constraints should only be applied on asynchronous clock domain crossing paths. When applied on an synchronous or intra-clock path, set\_data\_delay may cause Fitter optimizations to ignore or mishandle the effects of clock skew on the path's slack.

**Example Usage**

```
# Apply a 10ns max data delay on paths between two unrelated clocks
set_data_delay -from [get_clocks clkA] -to [get_clocks clkB] 10.000

# Apply a 2ns max data delay from an input port to any register
set_data_delay -from [get_ports in[*]] -to [get_registers *] 2.000

# Require net delay to be at most 90% of the period of the clock driving the inst9 register
set_data_delay -get_value_from_clock_period dst_clock_period -value_multiplier 0.9 -from
[get_clocks clk] -to [get_keepers inst9]

# Apply a 2ns max data delay for an input port only to nodes driven by
# the rising edge of clock CLK
set_data_delay -from [get_ports in[*]] -rise_to [get_clocks CLK] 2.000
```

**Return Value**

<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
TCL_ERROR	1	ERROR: The -add_launch_clock option can only be used when the -override option is used as well.
TCL_ERROR	1	ERROR: The option -value_multiplier must be a non-zero floating point number
TCL_ERROR	1	ERROR: The option -value_multiplier may only be used if -get_value_from_clock_period is used

### 3.1.28.18. set\_max\_skew (::quartus::sdc\_ext)

The following table displays information for the `set_max_skew` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sdc_ext</a> on page 375	
<b>Syntax</b>	<pre>set_max_skew [-h   -help] [-long_help] [-fall_from_clock &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-get_skew_value_from_clock_period &lt;src_clock_period dst_clock_period min_clock_period&gt;] [-rise_from_clock &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-skew_value_multiplier &lt;multiplier&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;] [&lt;skew&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-get_skew_value_from_clock_period <src_clock_period dst_clock_period min_clock_period>	Compute skew constraint as a multiple of the clock period
	-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-skew_value_multiplier <multiplier>	Value by which the clock period should be multiplied to compute skew requirement
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
<b>Description</b>	<p>Use the <code>set_max_skew</code> constraint to perform maximum allowable skew analysis between sets of registers or ports. In order to constrain skew across multiple paths, all such paths must be defined within a single <code>set_max_skew</code> constraint. <code>set_max_skew</code> timing constraint is not affected by <code>set_max_delay</code>, <code>set_min_delay</code>, and <code>set_multicycle_path</code>, but is affected by <code>set_false_path</code> and <code>set_clock_groups</code>: paths cut by a false path will not be analyzed for skew, and no two paths will be compared for skew if their clocks are exclusive to each other. However, paths whose clocks are asynchronous are still analyzed for skew. Legal values for the <code>-from</code> and <code>-to</code> options are collections of clocks, registers, ports, pins, cells or partitions in a design. Applying maximum skew constraints between clocks applies the constraint from all register or ports driven by the clock specified with the <code>-from</code> option to all registers or ports driven by the clock specified with the <code>-to</code> option. If pin names or collections are used, the <code>-from</code> value must be a clock pin and the <code>-to</code> value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells apply to all registers contained in the cell or driven by the clock pin. Similarly, <code>-to</code> and <code>-from</code> partition specifications apply to all registers in the specified partition. Max skew analysis includes data arrival times, clock arrival times, register micro parameters, clock uncertainty, on-die variation and ccpp removal. Use <code>-get_skew_value_from_clock_period</code> to set the skew requirement to be equal to the launching or latching clock period, or whichever of the two has a smaller period. If <code>-skew_value_multiplier</code> is used, the requirement is multiplied by that value. If this option is used, then the positional skew option may not be set. If the set of skew paths is clocked by more than one clock, the Timing Analyzer will use</p>	

**continued...**

	<p>the one with smallest period to compute the skew constraint. When this constraint is used, results of max skew analysis are displayed in the Report Max Skew (report_max_skew) report from the Timing Analyzer. Since skew is defined between two or more paths, no results are displayed if the -from/-from_clock and -to/-to_clock filters satisfy less than two paths.</p>																					
<b>Example Usage</b>	<pre># Constrain the skew on an input port to all registers it feeds set_max_skew -from [get_ports din] 0.200  # Constrain the skew on output bus dout[*] set_max_skew -to [get_ports dout\[*\]] 0.200  # Constrain skew to be less than 90% of the period of any clock in the source # register set set_max_skew -to [get_keepers inst1[*]] -get_skew_value_from_clock_period src_clock_period - skew_value_multiplier 0.900  # Report the results of max skew assignments report_max_skew -panel_name "Report Max Skew" -npaths 10 -detail path_only</pre>																					
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: &lt;string&gt; is not a valid argument for option -exclude. Available argument is to_clock</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Following options are missing required arguments: &lt;string&gt;</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: The option -skew_value_multiplier must be a non-zero floating point number</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: The option -skew_value_multiplier may only be used if -get_skew_value_from_clock_period is used</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: <string> is not a valid argument for option -exclude. Available argument is to_clock	TCL_ERROR	1	ERROR: Following options are missing required arguments: <string>	TCL_ERROR	1	ERROR: The option -skew_value_multiplier must be a non-zero floating point number	TCL_ERROR	1	ERROR: The option -skew_value_multiplier may only be used if -get_skew_value_from_clock_period is used
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>																				
TCL_OK	0	INFO: Operation successful																				
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.																				
TCL_ERROR	1	ERROR: <string> is not a valid argument for option -exclude. Available argument is to_clock																				
TCL_ERROR	1	ERROR: Following options are missing required arguments: <string>																				
TCL_ERROR	1	ERROR: The option -skew_value_multiplier must be a non-zero floating point number																				
TCL_ERROR	1	ERROR: The option -skew_value_multiplier may only be used if -get_skew_value_from_clock_period is used																				

### 3.1.28.19. `set_net_delay` (::quartus::sdc\_ext)

The following table displays information for the `set_net_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375	
<b>Syntax</b>	<pre>set_net_delay [-h   -help] [-long_help] -from &lt;names&gt; [-get_value_from_clock_period &lt;src_clock_period dst_clock_period min_clock_period max_clock_period&gt;] [-max] [-min] [-to &lt;names&gt;] [-value_multiplier &lt;multiplier&gt;] [&lt;delay&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-from <names>	Valid source pins, ports, registers or nets(string patterns are matched using Tcl string matching)
	-get_value_from_clock_period <src_clock_period dst_clock_period min_clock_period max_clock_period>	Compute net delay constraint as a multiple of the clock period
	-max	Specifies maximum delay
	-min	Specifies minimum delay
	-to <names>	Valid destination pins, ports, registers or nets (string patterns are matched using Tcl string matching)
	-value_multiplier <multiplier>	Value by which the clock period should be multiplied to compute net delay requirement

*continued...*

	<i>&lt;delay&gt;</i>	Required delay	
<b>Description</b>	Use the set_net_delay command to query the net delays and perform minimum or maximum timing analysis across nets. The -from and -to options can be string patterns or pin, port, register, or net collections. When pin or net collection is used, the collection should include output pins or nets. If the -to option is unused or if the -to filter is an "*" character, all the output pins and registers on timing netlist became valid destination points. When you use the -min option, slack is calculated by looking at the minimum delay on the edge. If you use -max option, slack is calculated with the maximum edge delay. Use -get_value_from_clock_period to set the net delay requirement to be equal to the launching or latching clock period, or whichever of the two has a smaller or larger period. If -value_multiplier is used, the requirement will be multiplied by that value. If the set of nets is clocked by more than one clock, the Timing Analyzer will use the one with smallest period to compute the constraint for a -max constraint, and the largest period for a -min constraint. If there are no clocks clocking the endpoints of the net (e.g. if the endpoints of the nets are not registers or constrained ports), then the net delay constraint will be ignored.		
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  # add min delay constraint set_net_delay -min 0.160 -from [get_pins inst9 combout] -to [get_pins * dataaf]  # add max delay constraint set_net_delay -max 0.500 -from inst8 combout  # this is same as the previous call set_net_delay -max 0.500 -from inst8 combout -to *  # Require net delay to be at most 90% of the period of the clock driving the inst9 register set_net_delay -max -get_value_from_clock_period dst_clock_period -value_multiplier 0.9 -from inst8 combout -to [get_keepers inst9]  update_timing_netlist  report_net_delay -panel "Net Delay"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: <string> is not a valid delay value
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: The option -value_multiplier must be a non-zero floating point number
	TCL_ERROR	1	ERROR: The option -value_multiplier may only be used if -get_value_from_clock_period is used

### 3.1.28.20. set\_scc\_mode (::quartus::sdc\_ext)

The following table displays information for the set\_scc\_mode Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375	
<b>Syntax</b>	set_scc_mode [-h   -help] [-long_help] [-size <size>] [-use_heuristic]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-size <size>	Maximum SCC loop size
	-use_heuristic	Always use heuristic for SCC processing

*continued...*

<b>Description</b>	Allows you to set maximum Strongly Connected Components (SCC) loop size or force the Timing Analyzer to always estimate delays through SCCs. When the Timing Analyzer encounters a loop of size greater than the specified maximum SCC loop size, it uses a heuristic which only estimates delays through the loop. If the loop is smaller than the maximum SCC loop size, a full processing of loops is performed unless the -use_heuristic option is used.		
<b>Example Usage</b>	<pre># Make the Timing Analyzer use normal processing for all loops # the size of which is less than or equal to 100. For loops of size # greater than 100, a runtime-saving heuristic will be used set_scc_mode -size 100  # Force the Timing Analyzer to use heuristic for all SCCs # disregarding their size set_scc_mode -use_heuristic</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.28.21. set\_time\_format (::quartus::sdc\_ext)

The following table displays information for the set\_time\_format Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	set_time_format [-h   -help] [-long_help] [-decimal_places <decimal_places>] [-unit <unit>]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-decimal_places <decimal_places>	Number of decimal places to use	
	-unit <unit>	Default time unit to use	
<b>Description</b>	Sets time format, including time unit and decimal places. Time units are assumed to be nanoseconds (ns) by default. The "-unit" option overrides the default time units. Legal time unit values are: ps, ns, us, ms. Time units are displayed with three decimal places by default. The "-decimal_places" option overrides the default number of decimal places to show. The smallest resolution of all times units is one picosecond (ps). Any additional specified precision will be truncated.		
<b>Example Usage</b>	<pre># Create two clocks with a clock period of 8 nanoseconds. create_clock -period 8.000 clk1  set_time_format -unit ps -decimal_places 0 create_clock -period 8000 clk2</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The default time unit can be set to ms, us, ns, or ps. Please specify one of these units instead.

### 3.1.28.22. set\_timing\_derate (::quartus::sdc\_ext)

The following table displays information for the set\_timing\_derate Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sdc_ext on page 375		
<b>Syntax</b>	set_timing_derate [-h   -help] [-long_help] [-cell_delay] [-early] [-late] [-net_delay] [-operating_conditions <operating_conditions>] <derate_value> [ <cells> ]		
<b>Arguments</b>	-h   -help	Short help	
<i>continued...</i>			

	-long_help	Long help with examples and possible return values	
	-cell_delay	Specifies that derating factors are only to apply to cell delays	
	-early	Specifies the minimum derating factor. This factor specifies how early the signal can arrive	
	-late	Specifies the maximum derating factor. This factor specifies how late the signal can arrive	
	-net_delay	Specifies that derating factors are only to apply to net delays	
	-operating_conditions <operating_conditions>	Operating conditions Tcl object	
	<derate_value>	Timing derate value	
	<cells>	List of cell type objects	
<b>Description</b>	Sets the global derate factors for the current design. The maximum and minimum delays of all timing arcs in the design are multiplied by the factors specified with the -late and -early options respectively. Only positive derate factors are allowed. If neither the -cell_delay nor -net_delay option is used, the derating factors apply to both cell and net delays. For net delay derates, the derate factor is applied to nets driven by matching cells. Specifying a derate value of less than 1.0 for the -late option or a derate value of greater than 1.0 for the -early option reduces delay pessimism, which might lead to optimistic results from timing analysis. The effect of set_timing_derate command is deferred until the next time update_timing_netlist is called. To reset derate factors to original values, use the reset_timing_derate command. This assignment is for timing analysis only, and is not considered during timing-driven compilation.		
<b>Example Usage</b>	<pre>set_timing_derate -early 0.9 [get_cells *] set_timing_derate -late 1.1 [get_cells *]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option -<string> has illegal value: <string>. Specify a legal option value.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29. ::quartus::sta

The following table displays information for the **::quartus::sta** Tcl package:

<b>Tcl Package and Version</b>	::quartus::sta 1.0
<b>Description</b>	This package contains the set of Tcl functions for obtaining information from the Timing Analyzer.
<b>Availability</b>	This package is loaded by default in the following executables:  quartus_fit quartus_sta
<b>Tcl Commands</b>	<a href="#">add_to_collection</a> ( <b>::quartus::sta</b> ) on page 394 <a href="#">check_timing</a> ( <b>::quartus::sta</b> ) on page 395 <a href="#">create_report_histogram</a> ( <b>::quartus::sta</b> ) on page 397 <a href="#">create_slack_histogram</a> ( <b>::quartus::sta</b> ) on page 398 <a href="#">create_timing_netlist</a> ( <b>::quartus::sta</b> ) on page 399 <a href="#">create_timing_summary</a> ( <b>::quartus::sta</b> ) on page 403 <a href="#">delete_sta_collection</a> ( <b>::quartus::sta</b> ) on page 404 <a href="#">delete_timing_netlist</a> ( <b>::quartus::sta</b> ) on page 405 <a href="#">enable_ccpp_removal</a> ( <b>::quartus::sta</b> ) on page 405 <a href="#">enable_sdc_extension_collections</a> ( <b>::quartus::sta</b> ) on page 406 <a href="#">get_available_operating_conditions</a> ( <b>::quartus::sta</b> ) on page 406 <a href="#">get_cell_info</a> ( <b>::quartus::sta</b> ) on page 407

*continued...*

```

get_clock_domain_info (::quartus::sta) on page 408
get_clock_fmax_info (::quartus::sta) on page 409
get_clock_info (::quartus::sta) on page 410
get_clock_pair_info (::quartus::sta) on page 411
get_datasheet (::quartus::sta) on page 412
get_default_sdc_file_names (::quartus::sta) on page 413
get_edge_info (::quartus::sta) on page 414
get_entity_instances (::quartus::sta) on page 415
get_min_pulse_width (::quartus::sta) on page 416
get_net_info (::quartus::sta) on page 416
get_node_info (::quartus::sta) on page 417
get_object_info (::quartus::sta) on page 418
get_operating_conditions (::quartus::sta) on page 420
get_operating_conditions_info (::quartus::sta) on page 420
get_partition_info (::quartus::sta) on page 421
get_path (::quartus::sta) on page 422
get_path_info (::quartus::sta) on page 423
get_pin_info (::quartus::sta) on page 426
get_point_info (::quartus::sta) on page 427
get_port_info (::quartus::sta) on page 429
get_register_info (::quartus::sta) on page 430
get_timing_paths (::quartus::sta) on page 431
import_sdc (::quartus::sta) on page 433
locate (::quartus::sta) on page 434
print_total_sdc_processing_time (::quartus::sta) on page 435
query_collection (::quartus::sta) on page 436
read_sdc (::quartus::sta) on page 436
register_delete_timing_netlist_callback (::quartus::sta) on page 437
remove_from_collection (::quartus::sta) on page 438
report_advanced_io_timing (::quartus::sta) on page 438
report_asynch_cdc (::quartus::sta) on page 439
report_bottleneck (::quartus::sta) on page 441
report_cdc_viewer (::quartus::sta) on page 442
report_clock_fmax_summary (::quartus::sta) on page 443
report_clock_network (::quartus::sta) on page 444
report_clock_transfers (::quartus::sta) on page 446
report_clocks (::quartus::sta) on page 447
report_datasheet (::quartus::sta) on page 447
report_ddr (::quartus::sta) on page 448
report_design_metrics (::quartus::sta) on page 449
report_exceptions (::quartus::sta) on page 452
report_ini_usage (::quartus::sta) on page 455
report_logic_depth (::quartus::sta) on page 456
report_max_clock_skew (::quartus::sta) on page 458
report_max_skew (::quartus::sta) on page 459
report_metastability (::quartus::sta) on page 460
report_min_pulse_width (::quartus::sta) on page 462
report_neighbor_paths (::quartus::sta) on page 463
report_net_delay (::quartus::sta) on page 466
report_net_timing (::quartus::sta) on page 467
report_partitions (::quartus::sta) on page 468
report_path (::quartus::sta) on page 469
report_pipelining_info (::quartus::sta) on page 471
report_register_spread (::quartus::sta) on page 472
report_reset_statistics (::quartus::sta) on page 473
report_retiming_restrictions (::quartus::sta) on page 474
report_route_net_of_interest (::quartus::sta) on page 474
report_rskm (::quartus::sta) on page 475
report_sdc (::quartus::sta) on page 476
report_skew (::quartus::sta) on page 477
report_tccs (::quartus::sta) on page 478
report_timing (::quartus::sta) on page 479
report_timing_by_source_files (::quartus::sta) on page 482
report_timing_tree (::quartus::sta) on page 484
report_ucp (::quartus::sta) on page 486
set_operating_conditions (::quartus::sta) on page 487
timing_netlist_exist (::quartus::sta) on page 488
update_timing_netlist (::quartus::sta) on page 488
use_timing_analyzer_styleEscaping (::quartus::sta) on page 489
write_sdc (::quartus::sta) on page 490

```

### **3.1.29.1. add\_to\_collection (::quartus::sta)**

The following table displays information for the add\_to\_collection Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	add_to_collection [-h   -help] [-long_help] <collection_obj_1> <collection_obj_2>	
<b>Arguments</b>	-h   -help	Short help

*continued...*

	<table border="1"> <tr> <td>-long_help</td><td>Long help with examples and possible return values</td></tr> <tr> <td>&lt;collection_obj_1&gt;</td><td>First object collection</td></tr> <tr> <td>&lt;collection_obj_2&gt;</td><td>Second object collection</td></tr> </table>	-long_help	Long help with examples and possible return values	<collection_obj_1>	First object collection	<collection_obj_2>	Second object collection			
-long_help	Long help with examples and possible return values									
<collection_obj_1>	First object collection									
<collection_obj_2>	Second object collection									
<b>Description</b>	This command takes two collections and returns a new collection that is a union of the two. The second collection is allowed to be a string, whereas the first has to be previously-created collection, either by passing any of the "get_" functions directly, or by passing a variable that contains a collection (see code examples for this command). If a collection is used for the second argument, the types in the second collection must be the same as or a subset of the types in the first collection. If the first collection consists of keepers, the second collection can only consist of keepers, registers or ports. If the first collection consists of partitions, the second collection can only consist of partitions or cells. If the first collection consists of nodes, the second collection can only consist of nodes, keepers, registers, ports, pins, nets or combinational nodes.									
<b>Example Usage</b>	<pre>set kprsl [get_keepers b*] set regsl [get_registers a*]  set regs_union [add_to_collection \$kprsl \$regsl] #or: set regs_union [add_to_collection [get_keepers b*] \$regsl] #or even: set regs_union [add_to_collection \$kprsl a*]  # - note that the last statement will actually add all keepers with name a* #   not only registers! (will add IOs with name a*, if any)  # Get the first 100 nodes in the collection. query_collection \$regs_union -limit 100</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Cannot find specified collection. Specify an existing collection.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Cannot find specified collection. Specify an existing collection.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Cannot find specified collection. Specify an existing collection.								

### 3.1.29.2. check\_timing (::quartus::sta)

The following table displays information for the `check_timing` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>check_timing [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-include &lt;check_list&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-include <check_list>	Checks to perform
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

*continued...*

<b>Description</b>	<p>Checks for problems in the design or problems with design constraints. The check_timing command performs a series of different checks based on user-specified variables and options. There is no default list of checks. Use the -include option to specify which checks to perform. You must precede check_timing with update_timing_netlist. The no_clock check reports whether registers have at least one clock at their clock pin, and that ports determined to be clocks have a clock assigned to them, and also checks that PLLs have a clock assignment. The multiple_clock check verifies that registers have at most one clock at their clock pin. (When multiple clocks reach a register clock pin, it is undefined which clock is used for analysis. The generated_clock check verifies that generated clocks are valid. Generated clocks must have a source that is triggered by a valid clock. The no_input_delay check verifies that every input port that is not determined to be a clock has an input delay assignment. The no_output_delay check verifies that every output port has an output delay constraint. The partial_input_delay check verifies that input delays are complete, and ensures that input delays have a rise-min, fall-min, rise-max, and fall-max portion set. The partial_output_delay check verifies that output delays are complete, and makes sure that output delays have a rise-min, fall-min, rise-max, and fall-max portion set. The io_min_max_delay_consistency check verifies that min delay values specified by set_input_delay or set_output_delay assignments are less than max delay values. The reference_pin check verifies that reference pins specified in set_input_delay and set_output_delay using the -reference_pin option are valid. A reference_pin is valid if the -clock option specified in the same set_input_delay/set_output_delay command matches the clock that is in the direct fanin of the reference_pin. Being in the direct fanin of the reference_pin means that there must be no keepers between the clock and the reference_pin. The latency_override check reports whether the clock latency set on a port or pin overrides the more generic clock latency set on a clock. Clock latency can be set on a clock, where the latency applies to all keepers clocked by the clock, whereas clock latency can also be set on a port or pin, where the latency applies to registers in the fanout of the port or pin. The loops check verifies that there are no strongly connected components in the netlist. These loops prevent a design from being properly analyzed. The loops check also reports if loops exist but were marked so that they would not be traversed. The latches check reports latches in the design and warns that latches may not be analyzed properly. For best results, change your design to remove latches whenever possible. The pos_neg_clock_domain check determines if any register is clocked by both the rising and falling edges of the same clock. If this scenario is necessary such as in a clock multiplexer, create two separate clocks that have similar settings and are assigned to the same node. The pll_cross_check checks the clocks that are assigned to a PLL against the PLL settings defined in design files. Inconsistent settings or an unmatched number of clocks associated with the PLL are reported to the user. The uncertainty check reports each clock-to-clock transfer that does not have a clock uncertainty assignment set between the two clocks. When a device family has derive_clock_uncertainty support, this report also checks if a user-defined set_clock_uncertainty assignment has a less than recommended clock uncertainty value. The virtual_clock check reports all unreferenced virtual clocks. It also reports if design does not have any virtual clock assignment. The partial_multicycle check ensures that each setup multicycle assignment has a corresponding hold multicycle assignment, and each hold multicycle assignment has a corresponding setup multicycle assignment. The multicycle_consistency check reports all the multicycle cases where a setup multicycle does not equal one greater than the hold multicycle. Hold multicycle assignments are usually one cycle less than setup multicycle assignments. The partial_min_max_delay check verifies that each minimum delay assignment has a corresponding maximum delay assignment, and vice versa. The clock_assignments_on_output_ports check reports all the clock assignments that have been applied to output ports. The input_delay_assigned_to_clock check verifies that no input delay value is set for a clock. Input delays set on clock ports are ignored because clock-as-data analysis takes precedence. The internal_io_delay check reports all the IO delays that have no specifications for -reference_pin and -source_latency_included, and -clock is a clock that is not assigned to a top level input or output port.</p>						
<b>Example Usage</b>	<pre># Constrain design create_clock -name clk -period 4.000 -waveform { 0.000 2.000 } [get_ports clk] set_input_delay -clock clk2 1.5 [get_ports in*] set_output_delay -clock clk 1.6 [get_ports out*] set_false_path -from [get_keepers in] -through [get_nets r1] -to [get_keepers out]  # Check if there were any problems check_timing -include {loops latches no_input_delay partial_input_delay}</pre>						
<b>Return Value</b>	<table border="1" data-bbox="421 1586 1406 1664"> <thead> <tr> <th data-bbox="421 1586 584 1622">Code Name</th><th data-bbox="584 1586 747 1622">Code</th><th data-bbox="747 1586 992 1622">String Return</th></tr> </thead> <tbody> <tr> <td data-bbox="421 1622 584 1664">TCL_OK</td><td data-bbox="584 1622 747 1664">0</td><td data-bbox="747 1622 992 1664">INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.29.3. create\_report\_histogram (::quartus::sta)

The following table displays information for the `create_report_histogram` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>create_report_histogram [-h   -help] [-long_help] [-append] [-color_div &lt;color_div&gt;] [-color_list &lt;color_list&gt;] [-file &lt;name&gt;] [-max_data &lt;max_data&gt;] [-min_data &lt;min_data&gt;] [-num_bins &lt;num_bins&gt;] [-panel_name &lt;name&gt;] [-stdout] [-x_label &lt;x_label&gt;] [-x_unit &lt;x_unit&gt;] [-y_label &lt;y_label&gt;] &lt;data&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-color_div <color_div>	Color divisions for the created histogram
	-color_list <color_list>	List of colors for painting the created histogram
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-max_data <max_data>	Maximum data value of the created histogram
	-min_data <min_data>	Minimum data value of the created histogram
	-num_bins <num_bins>	Number of bins
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
	-x_label <x_label>	Text label on x-axis
	-x_unit <x_unit>	Unit to be displayed on x-axis
	-y_label <y_label>	Text label on y-axis
	<data>	List of data to be analyzed
<b>Description</b>	Create a user defined histogram. Use <data> to specify the data entries to be displayed on the histogram. It can be a tcl list of either one of the following two formats or a mix of the two: {time_integer} or {time_integer number_count}, where time_integer is an integer possibly with a unit representing time (default unit is second), and number_count is a positive integer specifying number of entries (y value) of the corresponding time_integer. Use -num_bins to specify the number of bins, or the number of bars to be displayed on the histogram. Use -color_div and -color_list to specify the color of each bin. -color_div takes a tcl list of time_integers (see <data> above). Each entry in the list specifies the upper bound of each color division and therefore is forced to be a boundary of bins. -color_list takes a tcl list of colors. Each color in the list is used in the order specified and if less color is given than color divisions, the list will be re-used. For example, if specified "-color_div {-1 0 1} -color_list {red green}", then bins below -1 will be red, bins between -1 and 0 will be green, bins between 0 and 1 will be red again, and bins larger than 1 will be blue again. Possible choices of colors: black, blue, brown, green, grey, light_grey, orange, purple, red, white. Default -color_div is {0} and default -color_list is {red blue}. Use -max_data to specify the upper bound, i.e. largest number to be included in the histogram. Use -min_data to specify the lower bound, i.e. smallest number to be included in the histogram. Use -panel_name to specify the path and panel name of the created histogram. e.g. "-panel_name {Folder 1  Histogram 1}" will create a histogram named "Histogram 1" and put it in a folder with the name "Folder 1". Use -x_label to specify the text label on x_axis. Use -y_label to specify the text label on y_axis. Use -x_unit to specify a text unit to be attached to x_axis.	

*continued...*

<b>Example Usage</b>	<pre># create a path-based slack histogram project_open my_project create_timing_netlist read_sdc update_timing_netlist  # get path-based slack data in the format of a tcl list set data [list] set paths [get_timing_paths -setup -npaths 1000] foreach_in_collection path \$paths {     lappend data [get_path_info \$path -slack] }  # output data to histogram create_report_histogram \$data -panel_name {Path-based Slack Histogram} -num_bins 20 delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.

### 3.1.29.4. `create_slack_histogram` (::quartus::sta)

The following table displays information for the `create_slack_histogram` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>create_slack_histogram [-h   -help] [-long_help] [-all_edges] [-append] -clock_name &lt;name&gt; [-data_delay] [-file &lt;name&gt;] [-hold] [-max_slack &lt;max_slack&gt;] [-min_slack &lt;min_slack&gt;] [-num_bins &lt;num_bins&gt;] [-panel_name &lt;name&gt;] [-partition] [-recovery] [-removal] [-setup] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-all_edges	Consider slacks at all edges, not just at the endpoint nodes (results in increased memory consumption)
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-clock_name <name>	Name of the Clock Domain
	-data_delay	Data Delay Analysis (only applicable for setup and recovery analysis)
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-hold	Hold Analysis
	-max_slack <max_slack>	Maximum slack value of the created histogram
	-min_slack <min_slack>	Minimum slack value of the created histogram
	-num_bins <num_bins>	Number of bins
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-partition	Show slack count for each of the partition

*continued...*

	-recovery	Recovery Analysis	
	-removal	Removal Analysis	
	-setup	Setup Analysis	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Creates a slack histogram in the timing report for the specified clock domain "-clock_name," showing the number of timing edges within various ranges of slacks for a clock setup analysis. The histogram can be named using the "-panel_name" option. By default, only slack at the endpoint nodes of the timing netlist are considered. To include slack at all edges in the histogram, use the "-all_edges" option. This option will result in increased memory consumption. Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. If none is specified, setup analysis is used by default. Reports can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical interface ("panel_name"), or any combination of the three. The range of reported slack values can be controlled by specifying the "min_slack" and "max_slack" options. The number of bins (histogram bars) can also be specified using the "num_bins" option. Use the "-partition" to show more information about each partition. A path is in a partition if its starting point is in the partition. This option only works for "-panel".		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  # Create a slack histogram for clk1, defaulting to # the name "slack Histogram (clk1)" create_slack_histogram -clock_name clk1  # Create a slack histogram for clk2 named "MyHistogram" create_slack_histogram -clock_name clk2 -panel_name MyHistogram  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Clock node not found or specified. Check valid clocks
	TCL_ERROR	1	ERROR: Value <string> of option <string> is an invalid slack. Specify a valid slack value.
	TCL_ERROR	1	ERROR: The max_slack value is less than or equal to the min_slack value. Specify a max_slack value that is greater than the min_slack value.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Number of bins is 0. Specify a number greater than 0.

### 3.1.29.5. create\_timing\_netlist (::quartus::sta)

The following table displays information for the `create_timing_netlist` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393
<b>Syntax</b>	<code>create_timing_netlist [-h   -help] [-long_help] [-force_dat] [-grade &lt;c i m e&gt;] [-model &lt;fast slow&gt;] [-no_latch] [-post_map] [-post_syn] [-snapshot &lt;snapshot&gt;] [-speed &lt;speed&gt;] [-temperature &lt;value_in_C&gt;] [-voltage &lt;value_in_mV&gt;] [-zero_ic_delays] [ &lt;operating_conditions&gt; ]</code>

*continued...*

<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-force_dat</code>	Option to force delay annotation
	<code>-grade &lt;c i m e a&gt;</code>	Option to specify temperature grade
	<code>-model &lt;fast slow&gt;</code>	Option to specify timing model
	<code>-no_latch</code>	Option to disable the analysis of latches as synchronous elements
	<code>-post_map</code>	Option to perform timing analysis on the post-synthesis netlist
	<code>-post_syn</code>	Option to perform timing analysis on the post-synthesis netlist
	<code>-snapshot &lt;snapshot&gt;</code>	Snapshot of the design to load
	<code>-speed &lt;speed&gt;</code>	Speed grade
	<code>-temperature &lt;value_in_C&gt;</code>	Operating temperature
	<code>-voltage &lt;value_in_mV&gt;</code>	Operating voltage
	<code>-zero_ic_delays</code>	Option to set all IC delays to zero
	<code>&lt;operating_conditions&gt;</code>	Operating conditions Tcl object name string
<b>Description</b>	Creates the timing netlist by annotating the atom netlist with delay information using post-fitting results. Use the <code>-post_map</code> option to obtain post-synthesis results. In an incremental compilation flow, after Analysis and Synthesis, merge the partitions in your design using the <code>merge_partitions</code> Tcl command (or the <code>quartus_cdb</code> executable) to complete the creation of a post-synthesis netlist before you use the <code>-post_map</code> option to create a timing netlist. In Quartus Prime Pro edition, you can use the <code>-snapshot</code> option to specify which netlist you want to perform timing analysis on. The <code>create_timing_netlist</code> command skips delay annotation by default. Use <code>-force_dat</code> to rerun delay annotation. This is required if any delay annotation setting is changed in the Quartus Prime project revision (e.g. <code>OUTPUT_PIN_LOAD</code> ). Use " <code>-model fast</code> " to run the analysis using the fast corner delay models first. The <code>-temperature</code> , <code>-voltage</code> , and <code>-speed</code> , options are also available. See help for <code>set_operating_conditions</code> for details on these options. You can use <code>model</code> , <code>temperature</code> and <code>voltage</code> options to specify operating conditions while creating timing netlist (temperature and voltage options are not supported by all families). You can also set operating conditions by passing an operating conditions object name as a positional argument to <code>create_timing_netlist</code> command. After the timing netlist has been created, you can use <code>set_operating_conditions</code> command to change timing models without deleting and re-creating the timing netlist. Use the <code>-grade</code> option to analyze the design at a different temperature grade. This option is provided to support what-if analysis and is not recommended for final sign-off analysis. Use the <code>-no_latch</code> option to analyze latches as combinational loops instead of synchronous elements. Use the <code>-zero_ic_delays</code> option to set all IC delays in the netlist to zero.	
<b>Example Usage</b>	<pre> project_open my_top # Create timing netlist before calling # any report functions create_timing_netlist  # Read SDC and update timing read_sdc update_timing_netlist  # Ready to call report functions report_timing -npaths 1 -clock_setup  # The following command is optional delete_timing_netlist  project_close project_open my_top  # Report worst case period for -9 speed grade create_timing_netlist -speed 9 </pre>	

*continued...*

```

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_setup -clock_filter clk
delete_timing_netlist

# Report hold violation for fastest corner
# Use set_operating_conditions instead
create_timing_netlist -model fast

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# If Delay Annotation has been run for the fast corner
# Force Delay Annotation
create_timing_netlist -model fast -force_dat

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# Report worst case period for post-technology mapping netlist
create_timing_netlist -post_map

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_setup -clock_filter clk
delete_timing_netlist

project_close

```

Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: The TCL command <string> is not supported in quartus_fit. It is only supported in quartus_sta. Please use quartus_sta instead of quartus_fit if you want to use this TCL command.
TCL_ERROR	1		ERROR: Can't create timing netlist for device family <string>. Run Analysis and Synthesis (quartus_syn) using this device family as a value for the --family option before running the Timing Analyzer (create_timing_netlist).
TCL_ERROR	1		ERROR: The Fast Forward snapshot cannot be used for sign-off timing analysis. Please see the Fast Forward Timing Closure Recommendations report for performance estimates.
TCL_ERROR	1		ERROR: Can't run the Timing Analyzer (quartus_sta) -- Periphery placement (quartus_fit --plan) failed or was not run. Run I/O Assignment Analysis (quartus_fit --plan) successfully before running the Timing Analyzer (create_timing_netlist -post_map).
TCL_ERROR	1		ERROR: Invalid snapshot <string>. Available snapshot(s): <string>
TCL_ERROR	1		ERROR: Both the -temperature and -voltage options and their values are required.
TCL_ERROR	1		ERROR: Can't find active revision. Make sure there is an open, active revision name.
TCL_ERROR	1		ERROR: Could not find the <string> timing netlist on disk. Please run this fitter stage first, and make sure you have enabled Optimize Timing in Advanced Fitter Settings.

*continued...*

TCL_ERROR	1	ERROR: Option -no_latch is not supported for the current family.
TCL_ERROR	1	ERROR: Values entered did not match any valid operating conditions. Available operating conditions are: <string>
TCL_ERROR	1	ERROR: The <string> device family cannot perform automatic multi-corner timing analysis because the family does not support the set_operating_conditions command.
TCL_ERROR	1	ERROR: <string> Device family is not supported by the Timing Analyzer.
TCL_ERROR	1	ERROR: Illegal value: <string>. Specify an integer ranging from -999999999 to 999999999 for the option -voltage
TCL_ERROR	1	ERROR: The design has not been fully routed. If you want to perform Timing Analysis on an earlier netlist please choose which snapshot to load for analysis. Available snapshot(s): <string>
TCL_ERROR	1	ERROR: The design has not been fully routed and retimed to optimize timing. If you want to perform Timing Analysis on an earlier netlist please choose which snapshot to load for analysis. Available snapshot(s): <string>
TCL_ERROR	1	ERROR: Can't create timing netlist with -post_map option for device family <string>. Create timing netlist without the -post_map OPTION
TCL_ERROR	1	ERROR: Can't create timing netlist with -post_map option for device family <string>. Run I/O Assignment Analysis (quartus_fit --plan) successfully before running the Timing Analyzer (create_timing_netlist -post_map).
TCL_ERROR	1	ERROR: Use of --post_map and --post_fpp options are not allowed in hierarchical mode. You can use --snapshot to specify the database to be used for analysis
TCL_ERROR	1	ERROR: Can't run the Timing Analyzer (quartus_sta) --Fitter (quartus_fit) failed or was not run. Run the Fitter (quartus_fit) successfully before running the Timing Analyzer (create_timing_netlist).
TCL_ERROR	1	ERROR: Can't run the Timing Analyzer (quartus_sta) --Partition Merge (quartus_cdb --merge) failed or was not run. Run the Partition Merge (quartus_cdb --merge) successfully before running the Timing Analyzer (create_timing_netlist -post_map).
TCL_ERROR	1	ERROR: Can't run the Timing Analyzer (quartus_sta) --Analysis and Synthesis (quartus_syn) failed or was not run. Run Analysis and Synthesis (quartus_syn) successfully before running the Timing Analyzer (create_timing_netlist -post_map).
TCL_ERROR	1	ERROR: Delay annotation not run. Run delay annotation before running the Timing Analyzer (quartus_sta).
TCL_ERROR	1	ERROR: You can only specify the -snapshot option when Hierarchical Design is enabled.
TCL_ERROR	1	ERROR: Pre-fit timing analysis is not supported for the specified operating conditions. Please use the <string>.
TCL_ERROR	1	ERROR: The Synthesis snapshot cannot currently be used for timing analysis.
TCL_ERROR	1	ERROR: Timing netlist already exists. Delete the timing netlist before running this command.
TCL_ERROR	1	ERROR: Unsupported option: <string>.

### 3.1.29.6. create\_timing\_summary (::quartus::sta)

The following table displays information for the `create_timing_summary` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>create_timing_summary [-h   -help] [-long_help] [-append] [-data_delay] [-file &lt;name&gt;] [-hold] [-mpw] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-setup] [-split_by_corner] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-data_delay	Data Delay Analysis (only applicable for setup and recovery analysis)
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-hold	Hold Analysis
	-mpw	Minimum Pulse Width Analysis
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-recovery	Recovery Analysis
	-removal	Removal Analysis
	-setup	Setup Analysis (Default)
	-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
<b>Description</b>	Reports the worst-case Clock Setup and Clock Hold slacks and endpoint TNS (total negative slack) per clock domain. Total negative slack is the sum of all slacks less than zero for either destination registers or ports in the clock domain. The number of endpoints in the domain with negative slack is also shown. This command shows the worst-case slack for each clock domain. You right click in these reports to run more detailed reports like Histograms and Report Timing. By default, this command creates a Setup Summary. This command can also generate a Hold Summary (-hold), Recovery Summary (-recovery), Removal Summary (-removal), or Minimum Pulse Width Summary (-mpw). The report can be directed to the Tcl console (-stdout, default), a file (-file), the Timing Analyzer graphical interface (-panel_name), or any combination of the three.	
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first and process constraints create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Create Clock Domain Summary create_timing_summary -panel_name "Setup Summary" create_timing_summary -hold -panel_name "Hold Summary"  # The following command is optional</pre>	

*continued...*

	delete_timing_netlist project_close		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Panel name cannot contain repeated pipe characters. (  ).
	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.7. delete\_sta\_collection (::quartus::sta)

The following table displays information for the delete\_sta\_collection Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	delete_sta_collection [-h   -help] [-long_help] <collection>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<collection>	The collection to delete	
<b>Description</b>	This function is deprecated since sta will clean up the memory once the collection is out of scope. If you want to force a collection to go out of scope, use built-in tcl command 'unset'. Otherwise this function can remove the collection in cache.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set nodes [get_nodes Reg*]  # ... # do some work with the \$nodes collection # ...  # Delete the collection. delete_sta_collection \$nodes  # ... # do more work # ...  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Cannot find specified collection. Specify an existing collection.

### 3.1.29.8. delete\_timing\_netlist (::quartus::sta)

The following table displays information for the `delete_timing_netlist` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>delete_timing_netlist [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Use this command to delete a timing netlist previously created using <code>create_timing_netlist</code> . This should be done at the end of a script or before calling <code>create_timing_netlist</code> again using different options or after recompiling the design. Use the <code>set_operating_conditions</code> command instead of <code>delete_timing_netlist</code> and <code>create_timing_netlist</code> to change timing models. This avoids the cost of deleting and re-creating the timing netlist, and also preserves current timing assignments.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.29.9. enable\_ccpp\_removal (::quartus::sta)

The following table displays information for the `enable_ccpp_removal` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>enable_ccpp_removal [-h   -help] [-long_help] [-depth &lt;depth&gt;] [-off] [-on]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-depth &lt;depth&gt;</code>	maximum clock tree depth for ccpp removal	
	<code>-off</code>	Disable this setting.	
	<code>-on</code>	Enable this setting.	
<b>Description</b>	Enables (or disables) common clock path pessimism (CCPP) removal during slack computation. CCPP removal can improve timing results by removing min/max delay differences from common portions of clock paths. Enabling CCPP removal increases the time required to perform timing analysis. When specified, the optional depth parameter limits the clock tree depth used for CCPP removal. This is generally not applicable to FPGA compilations where the clock tree is fixed, but for large designs with potentially deep synthesized clock trees this can reduce outlier run time. When not specified, or when specified with a value of 0, the complete clock tree is used for CCPP removal (i.e. full clock-tree depth).		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc  # Report timing without CCPP removal report_timing  # Enable CCPP removal and re-report timing. enable_ccpp_removal report_timing</pre>		

*continued...*

	delete_timing_netlist project_close		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.10. enable\_sdc\_extension\_collections (::quartus::sta)

The following table displays information for the enable\_sdc\_extension\_collections Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	enable_sdc_extension_collections [-h   -help] [-long_help] [-off] [-on]		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-off		Disable this setting.
	-on		Enable this setting.
<b>Description</b>	Enable the support of SDC extension collections, such as keeper, register and node collections. When enable_sdc_extension_collections is not used, using these collections causes an error. Default to -on option.		
<b>Example Usage</b>	<pre>project_open top enable_sdc_extension_collections -on create_timing_netlist read_sdc  report_timing -to_clock clk1 delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Found an SDC extension Tcl collection creation command: <string>. Either switch to an SDC Tcl collection creation command or turn on enable_sdc_extension_collections.
	TCL_ERROR	1	ERROR: Found an SDC extension collection type: <string>. Either switch to an SDC collection type or turn on enable_sdc_extension_collections.
	TCL_ERROR	1	ERROR: Found timing netlist in memory. Delete timing netlist before use this command.

### 3.1.29.11. get\_available\_operating\_conditions (::quartus::sta)

The following table displays information for the get\_available\_operating\_conditions Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	get_available_operating_conditions [-h   -help] [-long_help] [-all] [-compile]		

*continued...*

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-all</code>	Returns all available operating conditions	
	<code>-compile</code>	Returns only the operating conditions that are critical to analyze in the inner loop compilation	
<b>Description</b>	Returns a Tcl collection of available operating conditions for the current device. The Tcl collection contains the most extreme operating conditions within a user-specified junction temperature range. Use the <code>-all</code> option to return all available operating conditions.		
<b>Example Usage</b>	<pre>#do report timing for different operating conditions foreach_in_collection op [get_available_operating_conditions] {     set_operating_conditions \$op     update_timing_netlist     report_timing }  #see detailed information about operating conditions foreach_in_collection op [get_available_operating_conditions] {     puts "Delay Model: [get_operating_conditions_info \$op -model]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: VID mode is not supported by part <string>.

### 3.1.29.12. get\_cell\_info (::quartus::sta)

The following table displays information for the `get_cell_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_cell_info [-h   -help] [-long_help] [-buried_nodes] [-buried_regs] [-in_pin_names] [-in_pins] [-location] [-name] [-out_pin_names] [-out_pins] [-pin_names] [-pins] [-type] [-wysiwyg_type] &lt;cell_object&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-buried_nodes</code>	Return a collection of buried node IDs
	<code>-buried_regs</code>	Return a collection of buried register IDs
	<code>-in_pin_names</code>	Return a list of input pin names
	<code>-in_pins</code>	Return a collection of input pin IDs
	<code>-location</code>	Return the atom location in device
	<code>-name</code>	Return the cell name
	<code>-out_pin_names</code>	Return a list of output pin names
	<code>-out_pins</code>	Return a collection of output pin IDs
	<code>-pin_names</code>	Return a list of input and output pin names
	<code>-pins</code>	Return a collection of input and output pin IDs

*continued...*

	-type	Return the cell type	
	-wysiwyg_type	Return the WYSIWYG type of the cell	
	<cell_object>	Cell object	
<b>Description</b>	Gets information about the specified cell (referenced by cell ID). You can obtain cell using the <code>get_cells</code> Tcl command. The "-type" option returns "cell". Options "-name", "-type", "-pin_name", "-in_pin_names", "-out_pin_names", "-pins", "-clock_pins", "-in_pins", "-out_pins", "-buried_nodes", "-buried_regs", "-location", and "-wysiwyg_type" are mutually exclusive.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set cells [get_cells] foreach_in_collection cell \$cells {     puts "[get_cell_info \$cell -name]: [get_cell_info \$cell -type]" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.13. `get_clock_domain_info` (::quartus::sta)

The following table displays information for the `get_clock_domain_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_clock_domain_info [-h   -help] [-long_help] [-data_delay] [-edge_slack] [-hold] [-mpw] [-recovery] [-removal] [-setup]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-data_delay	Data Delay Analysis (only applicable for setup and recovery analysis)
	-edge_slack	Compute edge TNS (this option may significantly increase memory consumption)
	-hold	Hold Analysis
	-mpw	Minimum Pulse Width Analysis
	-recovery	Recovery Analysis
	-removal	Removal Analysis
	-setup	Setup Analysis (Default)

*continued...*



<b>Description</b>	Similar to create_timing_summary, the get_clock_domain_info command returns a Tcl list of information about each active clock domain: the clock name, worst-case slack, endpoint TNS, number of endpoints in the domain with negative slack, and the timing model for which the worst-case slack is most critical. If the edge_slack option is specified, an extra entry for the edge TNS will be present, placed after the endpoint TNS. Computing the edge TNS may result in a significant increase in memory consumption. TNS is total negative slack, and it is the sum of all slacks less than zero for either destination registers or ports in the clock domain (endpoint TNS) or for all edges affecting the clock domain (edge TNS). By default, this command creates a Setup Summary. This command can also generate a Hold Summary (-hold), Recovery Summary (-recovery), Removal Summary (-removal), or Minimum Pulse Width Summary (-mpw).									
<b>Example Usage</b>	<pre>project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Get domain summary object set domain_list [get_clock_domain_info -setup] foreach domain \$domain_list {     set name [lindex \$domain 0]     set slack [lindex \$domain 1]     set keeper_tns [lindex \$domain 2]      puts "Clock \$name : Slack = \$slack , TNS = \$keeper_tns" }  # The following command is optional delete_timing_netlist  project_close</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
Code Name	Code	String Return								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.29.14. get\_clock\_fmax\_info (::quartus::sta)

The following table displays information for the get\_clock\_fmax\_info Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	get_clock_fmax_info [-h   -help] [-long_help]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
<b>Description</b>	Reports potential Fmax for every clock in the design, regardless of the user-specified clock periods. Fmax is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, Fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained. Restricted Fmax considers hold timing in addition to setup timing, as well as minimum pulse and minimum period restrictions. Similar to unrestricted Fmax, the restricted Fmax is computed as if the rising and falling edges of the clock are scaled along with Fmax, such that the duty cycle (in terms of a percentage) is maintained. Refer to hold timing reports (e.g., report_timing with the -hold option) or minimum pulse width reports (report_min_pulse_width) for details about specific paths, registers, or ports. This command is similar to report_clock_fmax_summary, except that it returns the results as a Tcl list for use in Tcl scripts. Each entry in the list represents one clock domain. Each entry is a Tcl list of the clock name, fmax (MHz), and restricted Fmax (MHz).	
<b>Example Usage</b>	<pre>project_open my_project  # Always create the netlist first create_timing_netlist</pre>	

*continued...*

```

read_sdc my_project.sdc
update_timing_netlist

# Get domain summary object
set domain_list [get_clock_fmax_info]
foreach domain $domain_list {
    set name [lindex $domain 0]
    set fmax [lindex $domain 1]
    set restricted_fmax [lindex $domain 2]

    puts "Clock $name : Fmax = $fmax (Restricted Fmax = $restricted_fmax)"
}

# The following command is optional
delete_timing_netlist

project_close
  
```

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.15. get\_clock\_info (::quartus::sta)

The following table displays information for the get\_clock\_info Tcl command:

Tcl Package and Version	Belongs to ::quartus::sta on page 393	
Syntax	get_clock_info [-h   -help] [-long_help] [-child_clocks] [-divide_by] [-duty_cycle] [-edge_shifts] [-edges] [-fall] [-is_inverted] [-latency] [-master_clock] [-master_clock_pin] [-max] [-min] [-multiply_by] [-name] [-nreg_neg] [-nreg_pos] [-offset] [-period] [-phase] [-rise] [-targets] [-type] [-waveform] <clk_object>	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-child_clocks	Returns a list of child clock names
	-divide_by	Return the frequency divider (to the base clock)
	-duty_cycle	Return the duty cycle
	-edge_shifts	Return a list of edge shifts that the specified edges are to undergo to yield the final generated clock waveform
	-edges	Return a list of integer representing edges from the source clock that are to form edges of the generated clock
	-fall	Return clock fall latency
	-is_inverted	Return a boolean value to indicate if the generated clock is inverted
	-latency	Return clock latency
	-master_clock	Return the master clock name
	-master_clock_pin	Return the master clock source pin
	-max	Return max clock latency
	-min	Return min clock latency
	-multiply_by	Return the frequency multiplier (to the base clock)

*continued...*

	-name	Return the clock name	
	-nreg_neg	Return number of registers negatively clocked by clock	
	-nreg_pos	Return number of registers positively clocked by clock	
	-offset	Return the clock offset	
	-period	Return the clock period	
	-phase	Return the clock phase	
	-rise	Return clock rise latency	
	-targets	Return the clock targets collection	
	-type	Return the clock type	
	-waveform	Return the waveform (rise time and fall time)	
	<clk_object>	Clock object	
<b>Description</b>	Returns information about the specified clock (referenced by clock ID). Clock IDs can be obtained by Tcl commands such as <code>get_clocks</code> . The "-type" option returns one of "base", "virtual_base", "generated", "virtual_generated". Options "-name", "-type", "-period", "-duty_cycle", "-waveform", "-edges", "-edge_shifts", "-multiply_by", "-divide_by", "-is_inverted", "-latency", "-master_clock", and "-targets" are mutually exclusive. The "-latency" option requires a specified "-max" or "-min" option as well as a "-rise" or "-fall" option.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set clocks [get_clocks] foreach_in_collection clk \$clocks {     puts "[get_clock_info \$clk -name]: [get_clock_info \$clk -period]" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.16. get\_clock\_pair\_info (::quartus::sta)

The following table displays information for the `get_clock_pair_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_clock_pair_info [-h   -help] [-long_help] [-fall_from &lt;clk_object&gt;] [-fall_to &lt;clk_object&gt;] [-false_path] [-from &lt;clk_object&gt;] [-hierarchy] [-hold] [-rise_from &lt;clk_object&gt;] [-rise_to &lt;clk_object&gt;] [-setup] [-to &lt;clk_object&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values

*continued...*

	-fall_from <clk_object>	Valid source clocks (string patterns are matched using Tcl string matching)	
	-fall_to <clk_object>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-false_path	Return a description of the satisfied false-path-type assignment applied between the "from" and "to" clocks, if any	
	-from <clk_object>	Valid source clocks (string patterns are matched using Tcl string matching)	
	-hierarchy	Return a description the hierarchical relationship between the "from" and "to" clocks	
	-hold	Return the hold analysis information if you use the "-false_path" option	
	-rise_from <clk_object>	Valid source clocks (string patterns are matched using Tcl string matching)	
	-rise_to <clk_object>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-setup	Return the setup analysis information if you use the "-false_path" option. The setup analysis relationship is returned by default	
	-to <clk_object>	Valid destination clocks (string patterns are matched using Tcl string matching)	
<b>Description</b>	The <code>get_clock_pair_info</code> command returns various clock information between two given clocks. If you specify the "-false_path" option, the command returns a description of the satisfied false path assignment between the "from" and "to" clocks, which includes clock groups. If you specify the "-hierarchy" option, the command returns a description of the clock hierarchy relationship between the two clocks, such as whether the "from" clock is a parent of the "to" clock. Use the "-from" option to specify the source clock that you want to query, and use the "-to" option to specify the destination clock that you want to query. When using the "-false_path" option, you can use the "-rise_from" option to specify a source clock's rising edge to report on, or use the "-fall_from" option to specify a source clock's falling edge to report on. Likewise, you can use the "-rise_to" option to specify a destination clock's rising edge, or use the "-fall_to" option to specify a destination clock's falling edge. You can also specify to retrieve either the setup or hold false path relationship using the "-setup" or "-hold" option respectively. By default, the setup relationship is reported.		
<b>Example Usage</b>	<pre> create_clock clkA -period 10 create_generated_clock clkB -source clkA -divide_by 2 create_generated_clock clkC -source clkB -divide_by 2  set_false_path -from clkA -to clkB -latency_insensitive set_clock_groups -group clkA -group clkC -asynchronous  get_clock_pair_info -from clkA -to clkB -false_path      -&gt; "false_path_latency_insensitive" get_clock_pair_info -from clkA -to clkC -false_path      -&gt; "clock_group_asynchronous" get_clock_pair_info -from clkA -to clkB -hierarchy       -&gt; "parent_child" get_clock_pair_info -from clkC -to clkA -hierarchy       -&gt; "descendant_ancestor" </pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.17. get\_datasheet (::quartus::sta)

The following table displays information for the `get_datasheet` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393
<b>Syntax</b>	<code>get_datasheet [-h   -help] [-long_help]</code>

*continued...*

<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> </ul>	Short help												
	<ul style="list-style-type: none"> <li>-long_help</li> </ul>	Long help with examples and possible return values												
<b>Description</b>	<p>This function returns a tcl collection which contains the datasheet report. Its format is as follows:</p> <pre>{ { tsu { &lt;tsu rise time&gt; &lt;tsu fall time&gt; &lt;input port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { th { &lt;th rise time&gt; &lt;th fall time&gt; &lt;input port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { tco { &lt;tco rise time&gt; &lt;tco fall time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { mintco { &lt;mintco rise time&gt; &lt;mintco fall time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { tpd { &lt;tpd rise-rise time&gt; &lt;tpd rise-fall time&gt; &lt;tpd fall-rise time&gt; &lt;tpd fall-fall time&gt; &lt;input port&gt; &lt;output port&gt; } } { mintpd { &lt;mintpd rise-rise time&gt; &lt;mintpd rise-fall time&gt; &lt;mintpd fall-rise time&gt; &lt;mintpd fall-fall time&gt; &lt;input port&gt; &lt;output port&gt; } } { tzx { &lt;tzx rise time&gt; &lt;tzx fall time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { mintzx { &lt;mintzx rise time&gt; &lt;mintzx fall time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { txz { &lt;txz time&gt; &lt;thz time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } { mintxz { &lt;mintxz time&gt; &lt;minthz time&gt; &lt;output port&gt; &lt;clock port&gt; &lt;clock edge&gt; &lt;clock reference&gt; } } } There are no options for this command, and the data returned is the same as from the report_datasheet command.</pre>													
<b>Example Usage</b>	<pre>project_open proj1 create_timing_netlist read_sdc update_timing_netlist  # get the datasheet collection set datasheet [get_datasheet]  # loop through contents of datasheet collection foreach i \$datasheet {     foreach j \$i {         foreach k \$j {             #             # extract individual items or             # manipulate as necessary             #         }     } }</pre>													
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Report database is not open</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open	
Code Name	Code	String Return												
TCL_OK	0	INFO: Operation successful												
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.												
TCL_ERROR	1	ERROR: Report database is not open												

### 3.1.29.18. `get_default_sdc_file_names` (::quartus::sta)

The following table displays information for the `get_default_sdc_file_names` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_default_sdc_file_names [-h   -help] [-long_help]</code>	
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> </ul>	Short help
	<ul style="list-style-type: none"> <li>-long_help</li> </ul>	Long help with examples and possible return values
<b>Description</b>	<p>Returns the default SDC file name(s) used by the Quartus Prime Compiler when doing timing-driven optimizations. Returns the value for the QSF variable SDC_FILE. If multiple assignments are found, return them as a list. If not specified, return &lt;revision_name&gt;.sdc.</p>	
<b>Example Usage</b>	<pre>project_new test create_timing_netlist foreach file [get_default_sdc_file_names] {     read_sdc \$file } update_timing_netlist</pre>	

*continued...*

	<pre>report_timing delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.19. get\_edge\_info (::quartus::sta)

The following table displays information for the `get_edge_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<code>get_edge_info [-h   -help] [-long_help] [-delay] [-delay_type] [-dst] [-ff] [-fr] [-hslp] [-is_disabled] [-max] [-min] [-name] [-rf] [-rr] [-src] [-type] [-unateness] &lt;edge_object&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-delay	Return the delay.
	-delay_type	Return the type of the delay (ic/cell).
	-dst	Return the destination node ID.
	-ff	Return the fall-to-fall delay
	-fr	Return the fall-to-rise delay
	-hslp	Return the HS/LP setting
	-is_disabled	Return whether the edge has been disabled, so should not be traversed through
	-max	Max delay
	-min	Min delay
	-name	Return the edge name
	-rf	Return the rise-to-fall delay
	-rr	Return the rise-to-rise delay
	-src	Return the source node ID
<b>Description</b>	Returns information about the specified edge (referenced by edge ID). Edge IDs can be obtained by Tcl commands such as <code>get_node_info &lt;node_id&gt; -synch_edges</code> . The "-type" and "-name" options exist only to keep the interface compliant with the <code>get_object_info</code> command. The "-type" option returns specific edge type as "synchronous", "asynchronous", "clock", or "combinational", while the "-name" option always returns an empty string. The "-delay" option returns the delay associated to the edge. Use -max, -min and -rr, -rf, -fr, -ff options to specify the type of returned delay. One of the -max, -min options must be specified. One of the -rr, -rf, -fr, -ff options must be specified. The -hslp option returns the HS/LP setting associated to the edge. The -unateness option returns the unateness associated to the edge. The -is_disabled option returns 1 if the edge should not be traversed through, and 0 if the edge can be traversed through. Disabled edges include edges in SCC loops and edges that the user has manually cut with the <code>set_disable_timing</code> command.	

*continued...*

<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set nodes [get_pins] foreach_in_collection node \$nodes {     set node_name [get_node_info -name \$node]     set edges [get_node_info \$node -fanout_edges]     foreach edge \$edges {         # Traverse to the fanout node         set dst_node [get_edge_info -dst \$edge]         set dst_name [get_node_info -name \$dst_node]         set delay_type [get_edge_info -delay_type \$edge]          set rr_delay [get_edge_info \$edge -max -delay -rr]         set rf_delay [get_edge_info \$edge -max -delay -rf]         set fr_delay [get_edge_info \$edge -max -delay -fr]         set ff_delay [get_edge_info \$edge -max -delay -ff]         puts "Max \$delay_type delay of edge \$edge, from \$node_name to \$dst_name: (RR:\$rr_delay RF:\$rf_delay FR:\$fr_delay FF:\$ff_delay)"     } } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option <string> is not allowed to be specified with option -<string>. Remove the disallowed option.
	TCL_ERROR	1	ERROR: Options <string> are exclusively allowed to be specified with option -<string>. Specify one of the allowed options.
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Option <string> is required to be specified with option -<string>. Specify a required option.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.20. get\_entity\_instances (::quartus::sta)

The following table displays information for the `get_entity_instances` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_entity_instances [-h   -help] [-long_help] [-nowarn] &lt;entity_name&gt;</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-nowarn</code>	Do not issue warning messages about missing entities
	<code>&lt;entity_name&gt;</code>	entity name
<b>Description</b>	Returns a tcl list of all hierarchical instance paths to a named given entity/module. This can be useful in SDC files which need to be applied to all instances of a module automatically.	

***continued...***

<b>Example Usage</b>	get_entity_instances entity_name		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

TCL\_OK      0      INFO: Operation successful

### 3.1.29.21. get\_min\_pulse\_width (::quartus::sta)

The following table displays information for the `get_min_pulse_width` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>get_min_pulse_width [-h   -help] [-long_help] [-nworst &lt;number&gt;] [-type &lt;all min_period clock_pulse&gt;] [&lt;targets&gt;]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-nworst <number>	Specifies the number of pulse width checks to report (default=1)	
	-type <all min_period clock_pulse>	Option to determine the minimum pulse width analysis type	
	<targets>	Registers or ports	
<b>Description</b>	This command returns a Tcl list which contains the minimum pulse width report. Its format is as follows: { { <slack>, <actual width>, <required width>, <pulse>, <clock>, <clock edge>, <target> } } Refer to help for the <code>report_min_pulse_width</code> command for help on the <code>-nworst</code> and <code>-targets</code> options.		
<b>Example Usage</b>	get_min_pulse_width		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.

### 3.1.29.22. get\_net\_info (::quartus::sta)

The following table displays information for the `get_net_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>get_net_info [-h   -help] [-long_help] [-name] [-pin] [-type] &lt;net_object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name	Return the net name	
	-pin	Return the pin ID of this net	
	-type	Return the net type.	
	<net_object>	Net object	
<b>Description</b>	Returns information about the specified net (referenced by net ID). Net ID's can be obtained by Tcl commands such as <code>get_nets</code> . The <code>"-type"</code> option returns "net". The options <code>"-name"</code> , <code>"-type"</code> , and <code>"-pin"</code> are mutually exclusive.		

*continued...*

<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set nets [get_nets] foreach_in_collection net \$nets {     puts [get_net_info \$net -name] }  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Option <string> is not allowed to be specified with option -<string>. Remove the disallowed option.
	TCL_ERROR	1	ERROR: Options <string> are exclusively allowed to be specified with option -<string>. Specify one of the allowed options.
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Option <string> is required to be specified with option -<string>. Specify a required option.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.23. get\_node\_info (::quartus::sta)

The following table displays information for the `get_node_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>get_node_info [-h   -help] [-long_help] [-asynch_edges] [-cell] [-clock_edges] [-fanout_asynch_edges] [-fanout_clock_edges] [-fanout_edges] [-fanout_synch_edges] [-location] [-name] [-synch_edges] [-type] &lt;node_object&gt;</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-asynch_edges	Return a list of asynchronous edge IDs
	-cell	Return the host cell
	-clock_edges	Return a list of clock edge IDs
	-fanout_asynch_edges	Return a list of asynchronous fanout edge IDs
	-fanout_clock_edges	Return a list of clock fanout edge IDs
	-fanout_edges	Return a list of fanout edge IDs
	-fanout_synch_edges	Return a list of synchronous fanout edge IDs
	-location	Return the atom location in device
	-name	Return the node name

*continued...*

	-synch_edges	Return a list of synchronous edge IDs	
	-type	Return the node type	
	<node_object>	Node object	
<b>Description</b>	Gets information about the specified node (referenced by node ID). Use Tcl commands such as get_nodes to obtain node IDs. The -type option returns "reg", "port", "pin", "net", or "comb". The -name, -type, -clock_edges, -synch_edges, -asynch_edges, -fanout_edges, -fanout_clock_edges, -fanout_synch_edges, -fanout_asynch_edges, -cell and -location options are mutually exclusive.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set registers [get_registers] foreach_in_collection reg \$registers {     puts "[get_node_info \$reg -name]: [get_node_info \$reg -type]" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.24. `get_object_info` (::quartus::sta)

The following table displays information for the `get_object_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>get_object_info [-h   -help] [-long_help] [-name] [-type] &lt;object&gt;</code>		
<b>Arguments</b>	-h   -help		Short help
	-long_help		Long help with examples and possible return values
	-name		Return the object name
	-type		Return the object type
	<object>		Object
<b>Description</b>	Gets information about the specified object (referenced by object ID). Object IDs can be obtained by Tcl commands such as <code>get_clocks</code> , <code>get_ports</code> , <code>get_cells</code> , and others. The -type option returns "clk", "reg", "port", "cell", "pin", "comb", "net", or "edge". The -name and -type options are mutually exclusive.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set ports [get_ports] foreach_in_collection port \$ports {     puts [get_object_info \$port -name] } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_ERROR	1	ERROR: Argument <string> is a collection ID that does not link to any collection. Specify a legal collection ID.
TCL_ERROR	1	ERROR: Argument <string> is an object ID that does not link to any object. Specify a valid object ID.
TCL_ERROR	1	ERROR: Argument <string> is an empty collection. Specify one that is a non-empty collection.
TCL_ERROR	1	ERROR: Argument <string> is not a valid object. Specify a valid object.
TCL_ERROR	1	ERROR: Argument <string> gives an empty collection. Specify one that gives a non-empty collection.
TCL_ERROR	1	ERROR: Argument <string> gives a collection with more than one object. Specify one that gives a collection with one object.
TCL_ERROR	1	ERROR: Argument <string> gives a collection that is not of <string> type. Specify one that gives a collection of required type.
TCL_ERROR	1	ERROR: Argument <string> gives an object that is not of <string> type. Specify one that gives an object of required type.
TCL_ERROR	1	ERROR: Argument <string> is a collection with more than one object. Specify a collection with one object.
TCL_ERROR	1	ERROR: Argument <string> is not a collection ID. Specify a legal collection ID.
TCL_ERROR	1	ERROR: Argument <string> is not an object ID. Specify a valid object ID.
TCL_ERROR	1	ERROR: Argument <string> is an object filter that matches more than one object. Specify a filter that matches only one object.
TCL_ERROR	1	ERROR: Argument <string> is an object filter that matches no objects. Specify one matches only one object.
TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.
TCL_ERROR	1	ERROR: Argument <string> is a collection that is not of <string> type. Specify a collection of required type.
TCL_ERROR	1	ERROR: Argument <string> is an object that is not of <string> type. Specify an object of required type.
TCL_ERROR	1	ERROR: Argument <string> is not <string> <string>. Specify an argument of the correct type.

### 3.1.29.25. `get_operating_conditions` (::quartus::sta)

The following table displays information for the `get_operating_conditions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>get_operating_conditions [-h   -help] [-long_help] [-for_analysis]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-for_analysis	Get the set of analysis corners instead of the set of reporting corners	
<b>Description</b>	Returns a list of the current operating conditions Tcl objects.		
<b>Example Usage</b>	<pre>puts "Delay Model : [get_operating_conditions_info [get_operating_conditions] -model]"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.26. `get_operating_conditions_info` (::quartus::sta)

The following table displays information for the `get_operating_conditions_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>get_operating_conditions_info [-h   -help] [-long_help] [-display_name] [-grade] [-is_hold_only] [-model] [-name] [-speed] [-temperature] [-voltage] &lt;operating_condition&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-display_name	Returns the operating conditions display name	
	-grade	Returns the temperature grade of the current device	
	-is_hold_only	Returns whether the operating conditions only support short-path analysis	
	-model	Returns the operating corner	
	-name	Returns the operating conditions Tcl_Obj name	
	-speed	Returns the speed grade of the current device	
	-temperature	Returns the operating temperature	
	-voltage	Returns the operating voltage	
<b>Description</b>	Returns information about the operating_conditions Tcl object.		

*continued...*

<b>Example Usage</b>	<pre>#see detailed information about operating conditions foreach_in_collection op [get_available_operating_conditions] {     puts "Delay Model: [get_operating_conditions_info \$op -model]" }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.27. get\_partition\_info (::quartus::sta)

The following table displays information for the `get_partition_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>get_partition_info [-h   -help] [-long_help] [-child] [-name] [-parent] [-type] &lt;partition_object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-child	Return child partition name(s)	
	-name	Return the partition name	
	-parent	Return parent partition name	
	-type	Return the partition type	
	<partition_object>	Partition object	
<b>Description</b>	Gets information about the specified partition (referenced by partition ID). Partition ID's can be obtained by Tcl commands such as <code>get_partitions</code> . The -name, -type, -parent, and -child options are mutually exclusive.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set partitions [get_partitions *] foreach_in_collection partition \$partitions {     puts "[get_partition_info \$partition -name]" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: This command is not supported in this version of the software.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.28. get\_path (::quartus::sta)

The following table displays information for the `get_path` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<pre>get_path [-h   -help] [-long_help] [-fall_from &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-from &lt;names&gt;] [-logic_depth] [-min_path] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-rise_from &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-show_routing] [-through &lt;names&gt;] [-to &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-logic_depth	Option to display the logic depth instead of path delay
	-min_path	Find the minimum delay path(s)
	-npaths <number>	Specifies the number of paths to report. The default value is 1 or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported (be wary that this may be slow)
	-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
	-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the longest delay path for each unique combination is displayed.
	-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-show_routing	Option to display detailed routing in the path
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
<b>Description</b>	<p>Returns a collection of path objects for the longest delay paths between arbitrary points in the netlist. This command behaves the same as the <code>report_path</code> command. However, instead of reporting the paths, it returns a Tcl collection of path objects. You can retrieve path object data using the <code>get_path_info</code> and <code>get_point_info</code> commands. Note that <code>get_path_info</code> does not provide any clock-</p>	

continued...

	related information, required points, or meaningful slack values, for paths represented by the path objects returned by this function. For help on the options shared with report_path, see help for the report_path command.																		
<b>Example Usage</b>	<pre># Define a few helper procedures to print out points # on a path, and the path itself  proc print_point { point } {     set total [ get_point_info \$point -total ]     set incr [ get_point_info \$point -incr ]     set node_id [ get_point_info \$point -node ]     set type [ get_point_info \$point -type ]     set rf [ get_point_info \$point -rise_fall ]     set node_name ""      if { \$node_id ne "" } {         set node_name [ get_node_info \$node_id -name ]     }      puts [format "%10s %8s %2s %-6s %s" \$total \$incr \$rf \$type \$node_name ] }  proc print_path { path } {     puts "Delay : [ get_path_info \$path -arrival_time]"     puts ""     puts [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]     puts "====="      foreach_in_collection pt [ get_path_info \$path -arrival_points ] {         print_point \$pt     } }  project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # And now simply iterate over the 10 longest delay paths, # printing each as we go. foreach_in_collection path [get_path -nworst 10] {     print_path \$path     puts "" }  delete_timing_netlist project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Option &lt;string&gt; has illegal value: &lt;string&gt;. Specify a legal option value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Collection type '&lt;string&gt;' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.	TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open
Code Name	Code	String Return																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.																	
TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.																	
TCL_ERROR	1	ERROR: Report database is not open																	

### 3.1.29.29. get\_path\_info (::quartus::sta)

The following table displays information for the get\_path\_info Tcl command:

Tcl Package and Version	Belongs to ::quartus::sta on page 393
Syntax	<pre>get_path_info [-h   -help] [-long_help] [-advanced] [-arrival_points] [-arrival_time] [-borrow_dst] [-borrow_src] [-clock_relationship] [-clock_skew] [-corner] [-data_delay] [-from] [-from_clock] [-from_clock_is_inverted] [-hold_end_multicycle] [-hold_start_multicycle] [-latch_time] [-launch_time] [-</pre>

*continued...*

	<pre>num_logic_levels] [-operating_conditions] [-required_points] [-required_time] [-setup_end_multicycle] [-setup_start_multicycle] [-slack] [-to] [-to_clock] [- to_clock_is_inverted] [-type] &lt;path_ref&gt;</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-advanced	Return clock pessimism due to advanced effects
	-arrival_points	Return a collection of point objects for the arrival path
	-arrival_time	Return the data arrival time for the path
	-borrow_dst	Return the time borrowed at destination (when the path ends at a level-sensitive latch)
	-borrow_src	Return the time borrowed at source (when the path starts at a level-sensitive latch)
	-clock_relationship	Return the clock relationship for the path
	-clock_skew	Return the clock skew for the path
	-corner	Returns a string indicating the operating conditions at which the path's delay values are found
	-data_delay	Return the data delay for the path
	-from	Return the source node ID
	-from_clock	Return the source clock ID
	-from_clock_is_inverted	Return 1 if the source clock is inverted, 0 otherwise
	-hold_end_multicycle	Return the hold end multicycle for the path
	-hold_start_multicycle	Return the hold start multicycle for the path
	-latch_time	Return the latch time for the path
	-launch_time	Return the launch time for the path
	-num_logic_levels	Return the number of logic levels on the path between the to node and from node
	-operating_conditions	Returns a string indicating the operating conditions at which the path's delay values are found
	-required_points	Return a collection of point objects for the required path
	-required_time	Return the data required time for the path
	-setup_end_multicycle	Return the setup end multicycle for the path
	-setup_start_multicycle	Return the setup start multicycle for the path
	-slack	Return the slack for the path
	-to	Return the destination node ID
	-to_clock	Return the destination clock ID
	-to_clock_is_inverted	Return 1 if the destination clock is inverted, 0 otherwise
	-type	Return the type of this path. Possible return values are: "setup", "hold", "recovery", "removal", "max_path", "min_path"
	<path_ref>	Path object

*continued...*

<b>Description</b>	<p>Returns information about the referenced timing path object. You can generate references to path objects with the <code>get_timing_paths</code> and <code>get_path</code> functions. The <code>-type</code> option returns one of the following types: "setup", "hold", "recovery", "removal", "max_path", or "min_path". The <code>-from</code> and <code>-to</code> options return the ID of the nodes at the start and end, respectively, of the arrival path. If there is no node, an empty string is returned. The <code>-from</code> node remains the same, regardless of the level of clock detail provided. It is always the first node clocked by the <code>-from</code> clock in the data arrival path. You can use the node ID with the <code>get_node_info</code> function to obtain additional information about the node. The <code>-from_clock</code> and <code>-to_clock</code> options return the ID of the launching and latching clocks, respectively. If there is no clock, an empty string is returned. You can obtain additional information about the clocks using the <code>get_clock_info</code> function. Path objects generated by <code>get_path</code> do not have clock information, required points, or meaningful slack values. The <code>-arrival_points</code> and <code>-required_points</code> options return a collection of point objects for the arrival and required paths, respectively. By iterating over the collection, and using the <code>get_point_info</code> function, you can obtain specific details about each portion of the path. If a path was created with additional clock detail, the elements of the clock path are included in each collection of points. The values for the <code>-from</code>, <code>-to</code>, and other options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details. When the path starts at a level-sensitive latch, the <code>-borrow_src</code> option may be used to get the time borrowed at the source. Similarly, when the path ends at a level-sensitive latch, <code>-borrow_dst</code> may be used to get the time borrowed at the destination. When these options are used with anything other than level-sensitive latches, zero is returned. For level-sensitive latches, when you use the <code>-launch_time</code> or <code>-latch_time</code> options, the times reported do not include time borrowed. The operating condition corresponding to all of a path's delay and time values can be found using the <code>-corner</code> option.</p>
<b>Example Usage</b>	<pre># Define a few helper procedures to print out points # on a path, and the path itself proc get_clock_string { path clk } {     set clk_str ""     set clk_id [ get_path_info \$path -\${clk}_clock ]     if { \$clk_id ne "" } {         set clk_str [ get_clock_info \$clk_id -name ]         if { [ get_path_info \$path -\${clk}_clock_is_inverted ] } {             append clk_str " (INVERTED)"         }     }     return \$clk_str }  proc print_point { point } {     set total [ get_point_info \$point -total ]     set incr [ get_point_info \$point -incr ]     set node_id [ get_point_info \$point -node ]     set type [ get_point_info \$point -type ]     set rf [ get_point_info \$point -rise_fall ]     set node_name ""      if { \$node_id ne "" } {         set node_name [ get_node_info \$node_id -name ]     }     puts [format "%10s %8s %2s %-6s %s" \$total \$incr \$rf \$type \$node_name ] }  proc print_path { path } {     puts "Slack : [ get_path_info \$path -slack]"     puts "To Clock : [ get_clock_string \$path to ]"     puts "From Clock : [ get_clock_string \$path from]"     puts ""     puts [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]     puts "===="     foreach_in_collection pt [ get_path_info \$path -arrival_points ] {         print_point \$pt     } }  project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # And now simply iterate over the 10 worst setup paths, printing each path foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {     print_path \$path     puts "</pre>

*continued...*

```

}
delete_timing_netlist
project_close

```

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.

### 3.1.29.30. get\_pin\_info (::quartus::sta)

The following table displays information for the `get_pin_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393				
<b>Syntax</b>	<code>get_pin_info [-h   -help] [-long_help] [-is_clock_pin] [-is_in_pin] [-is_out_pin] [-name] [-net] [-parent_cell] [-suffix] [-type] &lt;pin_object&gt;</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-is_clock_pin</code>	Return true if it is a clock pin, or false otherwise			
	<code>-is_in_pin</code>	Return true if it is an input pin, or false otherwise			
	<code>-is_out_pin</code>	Return true if it is an output pin, or false otherwise			
	<code>-name</code>	Return the pin name			
	<code>-net</code>	Return the net ID if this is an output pin			
	<code>-parent_cell</code>	Return the parent cell ID			
	<code>-suffix</code>	Return the suffix of the pin			
	<code>-type</code>	Return the pin type			
<b>Description</b>	Gets information about the specified pin (referenced by pin ID). Pin ID's can be obtained by Tcl commands such as <code>get_pins</code> . The <code>-type</code> option returns "pin". Options <code>-name</code> , <code>-type</code> , <code>-parent_cell</code> , <code>-net</code> , <code>-suffix</code> , <code>-is_clock_pin</code> , <code>-is_in_pin</code> and <code>-is_out_pin</code> are mutually exclusive.				
<b>Example Usage</b>	<pre> project_open chiptrip create_timing_netlist set pins [get_pins] foreach_in_collection pin \$pins {     set pin_name [get_pin_info \$pin -name]     set parent_cell [get_pin_info \$pin -parent_cell]     puts "Pin \$pin_name belongs to cell [get_cell_info -name \$parent_cell]" } delete_timing_netlist project_close </pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.		

*continued...*

TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.31. get\_point\_info (::quartus::sta)

The following table displays information for the get\_point\_info Tcl command:

Tcl Package and Version	Belongs to ::quartus::sta on page 393	
Syntax	get_point_info [-h   -help] [-long_help] [-edge] [-incremental_delay] [-location] [-node] [-number_of_fanout] [-rise_fall] [-total_delay] [-type] <point_ref>	
Arguments	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-edge	Return the edge ID for the edge associated with this point. If the point has no edge, this returns an empty string
	-incremental_delay	Return the incremental delay through this point
	-location	Return a string indicating the location of the point's node, if there is one, else an empty string
	-node	Return the node ID for the node associated with this point. If the point has no node, this returns an empty string
	-number_of_fanout	Return the number of fanout that this point has in the netlist
	-rise_fall	Return a string indicating the rise_fall type of this point. Return values are r, f, rr, rf, fr, ff, or an empty string for undefined
	-total_delay	Return the total delay of the path at this point. This includes the incremental delay for the point itself
	-type	Return a string indicating the type of the point
<point_ref>	Point object	
Description	<p>Returns information about the referenced timing point object. References to path objects can be generated using the get_path_info function. A point object is the equivalent of a row in a path in the output from report_timing. The -node option returns a node ID for the corresponding node in the path. For points that do not have a corresponding node (such as points for the lumped clock network delay, launch time, latch time, individual routing elements, etc.), the node ID is an empty string. A non-empty node ID can be used in conjunction with the get_node_info function to obtain additional information about the node. The -edge option returns an edge ID for the corresponding edge in the path. Only points of type "ic", "cell", and "comp" may have edges. For other point types, an empty string will be returned. A non-empty edge ID can be used in conjunction with the get_edge_info function to obtain additional information about the edge. The -total_delay option returns the total delay along the path, up to and including the current point. The -incremental_delay option returns the delay incurred by going through this point in the path. Both delays are formatted in terms of the current time units, excluding the unit string. The -number_of_fanout option returns the number of fanouts that the corresponding node has in the timing netlist. If there is no node for this point, the return value is 0. The -location option returns a string indicating the location of the corresponding node in the part. If there is no corresponding node, this returns an empty string. The -rise_fall option returns the transition type of this point. Possible values for -rise_fall are: Value Description ----- ----- (empty) Unknown transition r Rising input, falling output f Falling output rr Rising input, rising output rf Rising input, falling output fr Falling input, rising output ff Falling input, falling output The -type option returns a string indicating the type of delay that this point represents in the path.</p>	

*continued...*

	<p>Possible return values for -type are: Value Description -----</p> <p>----- borrow Time borrowed (for level-sensitive latches)</p> <p>cell Cell delay clknnet Lumped clock network delay clksrc Clock source. Used to ensure that the end-point of a clock segment is marked in the path when source latency is specified, or when the actual path cannot be found. comp PLL clock network compensation delay ic Interconnect delay iext External input delay latch Clock latch time launch Clock launch time loop Lumped combinational loop delay oext External output delay re Routing element (only for paths generated with the -show_routing option) srclat Source latency for a clock segment. This will appear if latency was specified between two clocks, or if a path could not be found between them. unc Clock uncertainty utco Register micro-Tco time utsu Register micro-Tsu time uth Register micro-Th time</p>									
<b>Example Usage</b>	<pre># Define a few helper procedures to print out points # on a path, and the path itself proc get_clock_string { path clk } {     set clk_str ""     set clk_id [ get_path_info \$path -\${clk}_clock ]      if { \$clk_id ne "" } {         set clk_str [ get_clock_info \$clk_id -name ]          if { [ get_path_info \$path -\${clk}_clock_is_inverted ] } {             append clk_str " (INVERTED)"         }     }      return \$clk_str }  proc print_point { point } {     set total      [ get_point_info \$point -total      ]     set incr       [ get_point_info \$point -incr       ]     set node_id    [ get_point_info \$point -node       ]     set type       [ get_point_info \$point -type       ]     set rf         [ get_point_info \$point -rise_fall]     set node_name  ""      if { \$node_id ne "" } {         set node_name [ get_node_info \$node_id -name ]     }      puts [format "%10s %8s %2s %-6s %s" \$total \$incr \$rf \$type \$node_name ] }  proc print_path { path } {     puts "Slack      : [ get_path_info \$path -slack]"     puts "To Clock   : [ get_clock_string \$path to ]"     puts "From Clock : [ get_clock_string \$path from]"     puts ""     puts [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]     puts "====="      foreach_in_collection pt [ get_path_info \$path -arrival_points ] {         print_point \$pt     } }  project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # And now simply iterate over the 10 worst setup paths, printing each path foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {     print_path \$path     puts "" }  delete_timing_netlist project_close</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Object with ID &lt;string&gt; is not an object of type &lt;string&gt;. Specify the ID of an object with the correct type.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.								

### 3.1.29.32. get\_port\_info (::quartus::sta)

The following table displays information for the `get_port_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <code>::quartus::sta</code> on page 393		
<b>Syntax</b>	<code>get_port_info [-h   -help] [-long_help] [-edge_rate] [-is inout_port] [-is input_port] [-is output_port] [-name] [-type] &lt;port_object&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-edge_rate</code>	Return the <code>edge_rate</code> value	
	<code>-is inout_port</code>	Return true if it is an inout port, or false otherwise	
	<code>-is input_port</code>	Return true if it is an input port, or false otherwise	
	<code>-is output_port</code>	Return true if it is an output port, or false otherwise	
	<code>-name</code>	Return the port name	
	<code>-type</code>	Return the port type	
	<code>&lt;port_object&gt;</code>	Port object	
<b>Description</b>	Returns information about the specified port (referenced by port ID). Port ID's can be obtained by Tcl commands such as <code>get_ports</code> . The <code>-type</code> option returns "port". The <code>-name</code> , <code>-type</code> , <code>-edge_rate</code> , <code>-is_input_port</code> , <code>-is_output_port</code> and <code>is_inout_port</code> options are mutually exclusive.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist set ports [get_ports] foreach_in_collection port \$ports {     set port_type ""     if {[get_port_info \$port -is inout_port]} {         set port_type "bidir"     } elseif {[get_port_info \$port -is input_port]} {         set port_type "in"     } else {         set port_type "out"     }     puts "[get_port_info \$port -name]: \$port_type" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Options are mutually exclusive: <string>. Specify only one of the these options.
	TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
	TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
	TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.33. get\_register\_info (::quartus::sta)

The following table displays information for the `get_register_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>get_register_info [-h   -help] [-long_help] [-asynch_edges] [-clock_edges] [-delay_type &lt;max_rise max_fall min_rise min_fall&gt;] [-fanout_edges] [-is_latch] [-name] [-related_pin &lt;related_pin_value&gt;] [-synch_edges] [-tch] [-tcl] [-tco] [-th] [-tmin] [-tsu] [-type] &lt;reg_object&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-asynch_edges	Return a list of asynchronous edge IDs	
	-clock_edges	Return a list of clock edge IDs	
	-delay_type <max_rise max_fall  min_rise min_fall>	Specify which type of micro delay to query	
	-fanout_edges	Return a list of fanout edge IDs	
	-is_latch	Return "1" if this is a latch node, or "0" otherwise	
	-name	Return the object name	
	-related_pin <related_pin_value>	Specify which register port you want the tsu/th/tco for	
	-synch_edges	Return a list of synchronous edge IDs	
	-tch	Return the Tch value	
	-tcl	Return the Tcl value	
	-tco	Return the Tco value	
	-th	Return the Th value	
<b>Description</b>	Gets information about the specified register (referenced by register ID). Register IDs can be obtained by Tcl commands such as <code>get_registers</code> . The <code>-type</code> option returns "reg". The <code>-name</code> , <code>-type</code> , <code>-tco</code> , <code>-tsu</code> , <code>-th</code> , <code>-tch</code> , <code>-tcl</code> , <code>-tmin</code> , <code>-clock_edges</code> , <code>-synch_edges</code> , <code>-asynch_edges</code> , <code>-fanout_edges</code> and <code>-is_latch</code> options are mutually exclusive.		
	<pre>project_open chiptrip create_timing_netlist set registers [get_registers] foreach_in_collection reg \$registers {     set name [get_register_info \$reg -name]     set tco [get_register_info \$reg -tco]     puts "Tco of \$name is \$tco" } delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: delay_type is used without specifying the related_pin. These two options must be used together.

*continued...*

TCL_ERROR	1	ERROR: related_pin is used without specifying the delay_type. These two options must be used together.
TCL_ERROR	1	ERROR: Object with ID <string> is not an object of type <string>. Specify the ID of an object with the correct type.
TCL_ERROR	1	ERROR: Cannot find object of ID <string>. Specify an existing object ID.
TCL_ERROR	1	ERROR: Unsupported object type: <string>. Specify a supported object type.

### 3.1.29.34. get\_timing\_paths (::quartus::sta)

The following table displays information for the get\_timing\_paths Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>get_timing_paths [-h   -help] [-long_help] [-data_delay] [-detail &lt;summary path_only path_and_clock full_path&gt;] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-false_path] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack limit&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-show_routing] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-data_delay	Report only paths that are covered by a data delay assignment
	-detail <summary path_only path_and_clock full_path>	Option to determine how much detail should be shown in the path report
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-false_path	Report only paths that are cut by a false path assignment
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-hold	Option to report clock hold paths
	-intra_clock	Only report paths whose launch and latch clock are the same

*continued...*

-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths
-show_routing	Option to display detailed routing in the path
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
<b>Description</b>	Get a collection of path objects for the worst-case paths. This command behaves the same as the report_timing command. However, instead of reporting the paths, it returns a Tcl collection of path objects. You can retrieve path object data using the get_path_info and get_point_info commands. For help on the options shared with report_timing, see the report_timing help page.
<b>Example Usage</b>	<pre># Define a few helper procedures to print out points # on a path, and the path itself proc get_clock_string { path clk } {     set clk_str ""     set clk_id [ get_path_info \$path -\${clk}_clock ]     if { \$clk_id ne "" } {         set clk_str [ get_clock_info \$clk_id -name ]         if { [ get_path_info \$path -\${clk}_clock_is_inverted ] } {             append clk_str " (INVERTED)"         }     }     return \$clk_str }  proc print_point { point } {     set total [ get_point_info \$point -total      ]     set incr  [ get_point_info \$point -incr     ]     set node_id [ get_point_info \$point -node   ]</pre>

*continued...*

```

set type      [ get_point_info $point -type      ]
set rf        [ get_point_info $point -rise_fall]
set node_name ""

if { $node_id ne "" } {
    set node_name [ get_node_info $node_id -name ]
}

puts [format "%10s %8s %2s %-6s %s" $total $incr $rf $type $node_name ]

proc print_path { path } {
    puts "Slack : [ get_path_info $path -slack]"
    puts "To Clock : [ get_clock_string $path to ]"
    puts "From Clock : [ get_clock_string $path from]"
    puts ""
    puts [format "%10s %8s %2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]
    puts "====="

    foreach_in_collection pt [ get_path_info $path -arrival_points ] {
        print_point $pt
    }
}

project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# And now simply iterate over the 10 worst setup paths, printing each path
foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {
    print_path $path
    puts ""
}

delete_timing_netlist
project_close

```

Return Value	Code Name	Code	String Return
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Option <string> has illegal value: <string>. Specify a legal option value.
TCL_ERROR	1		ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'
TCL_ERROR	1		ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
TCL_ERROR	1		ERROR: Report database is not open

### 3.1.29.35. import\_sdc (::quartus::sta)

The following table displays information for the `import_sdc` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>import_sdc [-h   -help] [-long_help]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
<b>Description</b>	Reads SDCs from synthesized database directly.	
<b>Example Usage</b>	<pre> project_new test create_timing_netlist  # Read SDC commands import_sdc  update_timing_netlist </pre>	

*continued...*

	<pre>report_timing delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.36. locate (::quartus::sta)

The following table displays information for the locate Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>locate [-h   -help] [-long_help] [-chip] [-classic_tmv] [-color &lt;black blue brown green grey light_grey orange purple red white&gt;] [-dpp] [-label &lt;label&gt;] [-no_duplicates] [-rpe] [-rtm] [-tmv] &lt;items&gt;</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-chip	Locate the object in the Chip Planner
	-classic_tmv	Locate the object in the Classic Technology Map Viewer
	-color <black blue brown green grey light_grey orange purple red white>	Specify the color to be used to identify the objects you are locating
	-dpp	Locate in the Design Partition Planner
	-label <label>	Specify a label used to identify the objects you are locating
	-no_duplicates	Do not locate duplicate objects
	-rpe	Locate in the Resource Property Editor
	-rtm	Locate in the Hyper-Retiming Viewer
	-tmv	Locate the object in the Technology Map Viewer
<b>Description</b>	<items>	Items to locate. Any collection or object (such as paths, points, nodes, nets, keepers, registers, etc) may be located by passing a reference to the corresponding collection or object.
	Locate an object from the Timing Analyzer in another Quartus Prime tool. With this command, one or more objects, or collections of objects, can be located in a supported Quartus tool from the Timing Analyzer. The destination must be specified with one of the following options: Option Destination Tool ===== ===== -chip Chip Planner -rpe Resource Property Editor -rtl RTL Viewer -classic_rtl Classic RTL Viewer -tmv Technology Map Viewer -classic_tmv Classic Technology Map Viewer -rtm Hyper-Retiming Viewer -dpp Design Partition Planner The -label option can be used to specify a label for the located objects. The -color command can be used to specify a color to be used to identify the located objects in the destination tool.	
<b>Example Usage</b>	<pre>proc prepare_design { } {     set sleep_for 2000      create_timing_netlist -risefall      post_message -type info "Give the GUI some time to catch up to the new netlist. Sleep for \$sleep_for ms"         after \$sleep_for</pre>	

*continued...*

```

        read_sdc
        update_timing_netlist
    }

prepare_design

# Locate all of the nodes in the longest ten paths
# into the Resource Property Editor
locate [get_path -npaths 10] -rpe

# Locate ten paths into the chip planner, labelling
# each one individually.
set path_col [get_timing_paths -npaths 10]
set path_id 0

foreach_in_collection path $path_col {
    incr path_id

    locate -label "Path #$path_id" $path -chip
}

# locate all keepers that begin with the letter t
# to the Tech Map Viewer
locate [get_keepers t*] -tmv

# locate all nodes that begin with the letter a
#
# The Timing Analyzer GUI will prompt the user for the
# tool to which the nodes should be located.
#
# Pause first to allow the previous locations to
# appear, as the dialog that pops up, to ask
# the user for a location, will block the rest
# of the GUI until cleared.

after 5000

post_message -type info "Interactive locate"
locate a*

```

<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
TCL_OK	0		INFO: Operation successful
TCL_ERROR	1		ERROR: Illegal color: <string>. Specify a color that is currently supported by the locate command.
TCL_ERROR	1		ERROR: An object or collection matching <string> could not be found, or was of a type not supported by the locate command.
TCL_ERROR	1		ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.37. print\_total\_sdc\_processing\_time (::quartus::sta)

The following table displays information for the print\_total\_sdc\_processing\_time Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	print_total_sdc_processing_time [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns the total processing time as a formatted string.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.38. query\_collection (::quartus::sta)

The following table displays information for the `query_collection` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>query_collection [-h   -help] [-long_help] [-all] [-limit &lt;limit_value&gt;] [-list_format] [-report_format] &lt;collection&gt;</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-all</code>	Return all the collection objects.	
	<code>-limit &lt;limit_value&gt;</code>	Set number of collection objects to return.	
	<code>-list_format</code>	Return collection objects in a list format.	
	<code>-report_format</code>	Return collection objects in a format of one element per line.	
	<code>&lt;collection&gt;</code>	Object collection	
<b>Description</b>	Query collection objects. Collections can be obtained by Tcl commands such as <code>get_clocks</code> , <code>get_ports</code> , <code>get_cells</code> . If neither the <code>-limit</code> nor the <code>-all</code> option is specified, then first 20 objects (if the collection has more than 20 objects) or all objects (if the collection has less than or equal to 20 objects) are returned.		
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist  set nodes [get_nodes Reg*] # Get the first 100 nodes in the collection. query_collection \$nodes -limit 100  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Cannot find specified collection. Specify an existing collection.

### 3.1.29.39. read\_sdc (::quartus::sta)

The following table displays information for the `read_sdc` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>read_sdc [-h   -help] [-long_help] [-hdl] [-instance &lt;instance_name&gt;] [-project_relative] [&lt;file_name&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-hdl</code>	Read SDC commands embedded in HDL	
	<code>-instance &lt;instance_name&gt;</code>	Name of the block for which we are reading the SDC file	
	<code>-project_relative</code>	If passing in a relative path to a file, interpret that path relative to the current project's directory instead of STA's current directory	
	<i>continued...</i>		

	<code>&lt;file_name&gt;</code>	Name of the SDC file	
<b>Description</b>	Reads an SDC file with all current constraints and exceptions. If an SDC file is specified, read_sdc only reads that SDC file. If the -hdl option is specified, read_sdc only reads SDC commands that were embedded in HDL. If no arguments are specified, read_sdc reads the default SDC files along with any SDC commands that were embedded in HDL. If one or more SDC_FILE assignments exists in the QSF, read_sdc reads all of them in order. Otherwise, read_sdc reads the file <revision>.sdc if it exists.		
<b>Example Usage</b>	<pre>project_new test create_timing_netlist  # Read SDC commands from test_constraints.sdc read_sdc test_constraints.sdc  # Read SDC commands embedded in HDL read_sdc -hdl  update_timing_netlist  report_timing  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Can't find file <string>
	TCL_ERROR	1	ERROR: The provided instance name does not exist in the current design.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.40. register\_delete\_timing\_netlist\_callback (::quartus::sta)

The following table displays information for the register\_delete\_timing\_netlist\_callback Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	register_delete_timing_netlist_callback [-h   -help] [-long_help] <body>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<body>	Body of the callback to run	
<b>Description</b>	Use this command to specify a TCL procedure that will run before the timing netlist is deleted. This command can be used to specify a procedure to run, like so: proc mycallback { # This is the body of the delete_timing_netlist callback. ... } register_delete_timing_netlist_callback mycallback Or, you may specify the procedure directly: register_delete_timing_netlist_callback { # This is the body of the delete_timing_netlist callback. ... }		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.29.41. remove\_from\_collection (::quartus::sta)

The following table displays information for the `remove_from_collection` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>remove_from_collection [-h   -help] [-long_help] &lt;base_collection&gt; &lt;items_to_remove&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	<base_collection>	Collection to remove items from	
	<items_to_remove>	Items to be removed from base_collection	
<b>Description</b>	This command takes two collections and returns a new collection that is the difference of the two, effectively the second collection subtracted from the first collection. The second collection can be a string, but the first has to be previously-created collection: either by passing any of the "get_" functions directly, or by passing a variable that contains a collection (see code examples for this command). If a collection is used for the second argument, the types in the second collection must be the same as or a subset of the types in the first collection. If the first collection consists of keepers, the second collection can only consist of keepers, registers or ports. If the first collection consists of partitions, the second collection can only consist of partitions or cells. If the first collection consists of nodes, the second collection can only consist of nodes, keepers, registers, ports, pins, nets or combinational nodes.		
<b>Example Usage</b>	<pre>set a_keepers [get_keepers a*] set a1_regs [get_registers a1*]  set keepers_without_a1 [remove_from_collection \$a_keepers \$a1_regs]  #or: set keepers_without_a1 [remove_from_collection \$a_keepers [get_registers a1*]]  #or even: set keepers_without_a1 [remove_from_collection \$a_keepers a1*]  # - note that the last statement will actually remove all keepers with name a1* #   not only registers! (will remove IOs with name a1*, if any)  # Get the first 100 nodes in the collection. query_collection \$keepers_without_a1 -limit 100</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Cannot find specified collection. Specify an existing collection.

### 3.1.29.42. report\_advanced\_io\_timing (::quartus::sta)

The following table displays information for the `report_advanced_io_timing` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>report_advanced_io_timing [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	

*continued...*

	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	This command creates a report containing all of the relevant signal integrity measurements computed during I/O buffer simulation. You must perform delay annotation with Advanced I/O Timing enabled before using this command. This option can be enabled from the Timing Analyzer Page of the Settings dialog box.		
<b>Example Usage</b>	<pre>project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Create "Advanced I/O Timing" report panel report_advanced_io_timing -panel_name "Advanced I/O Timing"  # The following command is optional delete_timing_netlist  project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.43. report\_asynch\_cdc (::quartus::sta)

The following table displays information for the `report_asynch_cdc` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_asynch_cdc [-h   -help] [-long_help] [-append] [-detail &lt;summary full&gt;] [-fall_from_clock &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-intra_clock] [-multi_bit_cdc] [-nentries &lt;number&gt;] [-panel_name &lt;name&gt;] [-reset_cdc] [-rise_from_clock &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-single_bit_cdc] [-stdout] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	<code>-detail &lt;summary full&gt;</code>	Option to specify how much detail should be shown in the CDC report
	<code>-fall_from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
	<code>-fall_to_clock &lt;names&gt;</code>	Valid destination clocks (string patterns are matched using Tcl string matching)

*continued...*

	<b>-file &lt;name&gt;</b>	Sends the results to an ASCII or HTML file. Depending on the extension	
	<b>-from &lt;names&gt;</b>	Valid sources (string patterns are matched using Tcl string matching)	
	<b>-from_clock &lt;names&gt;</b>	Valid source clocks (string patterns are matched using Tcl string matching)	
	<b>-intra_clock</b>	Only report paths whose launch and latch clock are the same	
	<b>-multi_bit_cdc</b>	Report multi-bit CDC topologies found in the design	
	<b>-nentries &lt;number&gt;</b>	Display up to this number of entries per CDC topology. Only applicable in full detail mode	
	<b>-panel_name &lt;name&gt;</b>	Sends the results to the panel and specifies the name of the new panel	
	<b>-reset_cdc</b>	Report reset CDC topologies found in the design	
	<b>-rise_from_clock &lt;names&gt;</b>	Valid source clocks (string patterns are matched using Tcl string matching)	
	<b>-rise_to_clock &lt;names&gt;</b>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	<b>-single_bit_cdc</b>	Report single-bit CDC topologies found in the design	
	<b>-stdout</b>	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	<b>-to &lt;names&gt;</b>	Valid destinations (string patterns are matched using Tcl string matching)	
	<b>-to_clock &lt;names&gt;</b>	Valid destination clocks (string patterns are matched using Tcl string matching)	
<b>Description</b>	This report displays all Clock-Domain-Crossings (CDC) between asynchronous clocks in a design. The report can be directed to the Tcl console ("stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. By default, this command reports all CDC's in a design. You can limit the analysis performed by this command to specific CDC source and destination nodes, using the "-from" and "-to" options. The analysis can also be limited using clocks. Specify the CDC's source and destination clocks using the "-from_clock" and "-to_clock" options. Alternatively, specify edges of the clock using "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. To limit the report to only display specific categories of CDC's, use "-multi_bit_cdc" to report multi-bit CDC buses, "-single_bit_cdc" to report single-bit CDC synchronizers, and "-reset_cdc" to report asynchronous reset topologies. By default, all three CDC categories are reported. Use the "-nentries" option to limit the number of CDC's displayed per CDC topology type. The topology types fall into one of the three categories above. Run the report to see all the topology types that are supported. Use the "-detail" option to specify the desired level of reporting detail. "summary" generates a table listing only the number of CDC topologies recognized per category and the total number of crossings for that category. "full" reports every recognized CDC under each topology type, and is the default behaviour. In the full report, you can click on each individual CDC in the Timing Analyzer graphical user interface to view its statistics.		
<b>Example Usage</b>	<pre># Report all bus CDC's, up to 10 buses per CDC topology type report_asynch_cdc -buses -nentries 10  # Report all CDC's from clkA to clkB report_asynch_cdc -from_clock clkA -to_clock clkB  # Report all synchronizer chains whose source node contains "transfer" report_asynch_cdc -from *transfer* -synch_chains</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: The current report cannot be run in XML mode. Use create_timing_netlist to create a non-XML timing netlist.

### 3.1.29.44. report\_bottleneck (::quartus::sta)

The following table displays information for the report\_bottleneck Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	report_bottleneck [-h   -help] [-long_help] [-cmetric <cmetric_name>] [-details] [-metric <default tns num_paths num_fpaths num_fanins num_fanouts>] [-nworst <number>] [-panel_name <panel_name>] [-stdout] [ <paths> ]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-cmetric <cmetric_name>	Custom metric function to evaluate individual nodes
	-details	Show the detailed information (number of failing edges, number of fanins, etc)
	-metric <default tns num_paths num_fpaths num_fanins num_fanouts>	Indicate the metric to use to rate individual nodes
	-nworst <number>	Specifies the maximum number of nodes to report. If unspecified, there is no limit
	-panel_name <panel_name>	Sends the results to the panel and specifies the name of the new panel
	-stdout	Output the result onto stdout
	<paths>	Paths to be analyzed
<b>Description</b>	<p>Reports bottleneck nodes in a design based on user-specified criteria for rating each node. The following considerations are pre-defined num_fpaths: (default) returns the number of paths that fail timing through the node. num_fanins: returns the number of fanin edges from the node. num_fanouts: returns the number of fanout edges from the node. num_paths: returns the number of paths through the node. tns: returns the total negative slack of all the paths through the node. The paths to be analyzed can be specified by passing the result of any get_timing_paths call as the last argument to report_bottleneck. If no paths are specified, report_bottleneck analyzes the worst 1000 setup paths in the design. You can also create your own custom criteria for evaluating nodes based on the combination of the number of fanouts, fanins, failing paths, and total paths. To use custom criteria, do the following: 1. Create a Tcl procedure that takes one argument, "arg", for example. 2. Use "upvar \$arg metric" in the procedure. 3. Calculate the rating based on \$metric(tns), \$metric(num_fanouts), \$metric(num_fanins), and \$metric(num_fpaths). 4. Return the rating with "return \$rating". 5. Pass the name of your custom criteria procedure to report_bottleneck using the -cmetric option. Reports can be directed to the Tcl console (-stdout), the Timing Analyzer graphical interface (-panel), or a combination of the two.</p>	
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  # use the worst 500 hold paths set paths [ get_timing_paths -npaths 500 -hold ] report_bottleneck -metric default -panel "Timing Analysis Bottleneck Report - Default Metric" \$paths report_bottleneck -metric tns -panel "Timing Analysis Bottleneck Report - TNS" \$paths report_bottleneck -metric num_paths -panel "Timing Analysis Bottleneck Report - Number of Paths" \$paths report_bottleneck -metric num_fpaths -panel "Timing Analysis Bottleneck Report - Number of Failing Paths" \$paths report_bottleneck -metric num_fanouts -panel "Timing Analysis Bottleneck Report - Number of Fanouts" \$paths  # create custom metric and use the worst 2000 setup paths proc report_bottleneck_custom_metric {arg} {     # Description: use the number of fanins as the custom metric.     upvar \$arg metric     set rating \$metric(num_fanins)     return \$rating }</pre>	

*continued...*

	<pre>set paths [ get_timing_paths -npaths 2000 -setup ] report_bottleneck -cmetric report_bottleneck_custom_metric -panel "Timing Analysis Bottleneck Report - Custom" \$paths</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.45. report\_cdc\_viewer (::quartus::sta)

The following table displays information for the report\_cdc\_viewer Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_cdc_viewer [-h   -help] [-long_help] [-append] [-clock_groups] [-file &lt;name&gt;] [-from_clock &lt;names&gt;] [-fully_cut] [-hierarchy] [-hold] [-inactive] [-less_than_slack &lt;slack limit&gt;] [-list] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-setup] [-show_empty] [-show_non_crossing] [-stdout] [-summary] [-timed] [-to_clock &lt;names&gt;] [-tree]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-clock_groups	Show transfers cut by clock groups
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fully_cut	Include transfers where all paths are cut by false path assignments
	-hierarchy	When writing to a panel or a non-list file, show child clocks as nested within their parent clocks
	-hold	Option to report clock hold paths
	-inactive	Show transfers between inactive clocks
	-less_than_slack <slack limit>	Ignore paths with slack values greater or equal to the specified limit
	-list	When writing to file, output all clock transfers in a list instead of a grid
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-recovery	Option to report recovery paths
	-removal	Option to report removal paths
	-setup	Option to report clock setup paths
	-show_empty	Include all clocks in the report, even ones that launch/latch no paths
	-show_non_crossing	Include transfers that do not cross a clock domain (i.e. transfers to/from the same clock)

*continued...*

	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-summary	Do not display slack information	
	-timed	Include all timed transfers (i.e. those not fully cut by false paths or clock groups)	
	-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-tree	Alias for the -hierarchy option	
<b>Description</b>	Generates a Clock Domain Crossing Viewer (CDC Viewer) report. It displays all clock transfers (i.e., data paths between one clock domain and another clock domain) in a design, as well as data on each transfer: the number of uncut and cut paths in the transfer, the worst-case and total-negative slack of the transfer, and the tightest setup/hold/removal/recovery relationship between the clocks in the transfer. The report indicates what clock transfers are cut ("false paths") by set_clock_groups or clock-to-clock set_false_path commands, and which clock transfers are ignored due to clocks marked as inactive by set_active_clocks. The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. When directed to a panel or a file without the -list option, the CDC Viewer report will be a grid, where each cell represents paths transferring between a source and destination clock. When directed to a panel or an HTML file, the grid will be color-coded to highlight passing, failing, cut, and inactive transfers, as well as clock groups. When directed to the Tcl console or a file with the -list option, a list of clock transfers will be reported. The -setup, -hold, -recovery, and -removal options determine the analysis type of the report, particularly the reporting of false_paths that apply to only one analysis type. If you do not specify any of these options, a report is generated for each analysis. By default, the only transfers that will be shown in the report are ones that participate in clock domain crossings. This means that transfers to/from the same clock will not be shown, even if they fail timing. To show all transfers, use the -show_all option. The generated report can be customized by a variety of options. Refer to the help text of those options for more information.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist -skip_dat report_cdc_viewer -panel_name delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Netlist must be updated. Run update_timing_netlist

### 3.1.29.46. report\_clock\_fmax\_summary (::quartus::sta)

The following table displays information for the `report_clock_fmax_summary` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_clock_fmax_summary [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-split_by_corner] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.

*continued...*

	<ul style="list-style-type: none"> <li>-file &lt;name&gt;</li> <li>-panel_name &lt;name&gt;</li> <li>-split_by_corner</li> <li>-stdout</li> </ul>	Sends the results to an ASCII or HTML file. Depending on the extension  Sends the results to the panel and specifies the name of the new panel  When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.  Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.									
<b>Description</b>	Reports potential fmax for every clock in the design, regardless of the user-specified clock periods. Fmax is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained. Restricted fmax considers hold timing in addition to setup timing, as well as minimum pulse and minimum period restrictions. Similar to unrestricted fmax, the restricted fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained. The "Note" column reports which analyses restricted fmax. Refer to hold timing reports (e.g., report_timing with the -hold option) or minimum pulse width reports generated by the report_min_pulse_width command for details of specific paths, registers, or ports.										
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Output results in the form of messages report_clock_fmax_summary # Create "Fmax" report panel report_clock_fmax_summary -panel_name Fmax # Report both with report panel and messages report_clock_fmax_summary -panel_name Fmax -stdout  # The following command is optional delete_timing_netlist  project_close</pre>										
<b>Return Value</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> </tbody> </table>		Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
Code Name	Code	String Return									
TCL_OK	0	INFO: Operation successful									
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.									

### 3.1.29.47. report\_clock\_network (::quartus::sta)

The following table displays information for the report\_clock\_network Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_clock_network [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-include_non_clock_paths] [-initial_depth &lt;number&gt;] [-panel_name &lt;name&gt;] [-show_full_paths] [-stdout] [-target &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values

*continued...*

	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-include_non_clock_paths	Show paths that are potentially in the clock network. These paths terminate at a register's clock pin but do not start at a clock source.	
	-initial_depth <number>	Initial clock network depth to report.	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-show_full_paths	Show the full clock paths - do not reduce the table size by condensing multiple trivial nodes into one row.	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-target <names>	Valid nodes in the clock network or clocks (string patterns are matched using Tcl string matching).	
<b>Description</b>	The clock network report shows the netlist topology of clock paths that make up the clock network in a design. It allows the user to track a clock signal from its source, through transformations such as PLL's, to the loads that the clock drives. It also reveals clock relationships by indicating nodes where generated clocks are defined. The report can be directed to the Tcl console ("stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("panel_name"), or any combination of the three. Use the "-target" option to specify report targets. These can be clocks in the design or nodes on the clock network, for example PLL outputs. If you specify clocks as targets, the clock network report will use the clock's target nodes as report targets. The clock network report then displays all nodes in the clock network that are in the fanin and fanout cones of the report targets. If no report target is specified, all clock target nodes in the design are used by default. Each row in the report may include one node or multiple trivial nodes with singular fanin and fanout edge. To disable the behaviour of collapsing down trivial nodes into one row, use the "-show_full_paths" option. When running the clock network report in the Timing Analyzer GUI, rows in the main table corresponding to report target nodes are highlighted in light blue. You may click on each row of the table to view information such as clock frequencies, relationships, and why this node belongs in the clock network. You may right-click on any row to locate the node in other Quartus tools such as RTL Viewer. Use the "-initial_depth" option to reduce the height of the report table in the Timing Analyzer GUI. Rows in the main table that are deeper than the set initial depth will be collapsed by default, but can be manually expanded. If this option is not set, the report determines an appropriate initial depth to use. Use the "-include_non_clock_paths" option if a node that you expect to be in the clock network cannot be found. This option shows nodes that lead into a register's clock pin, but are not downstream of any known clock source. Furthermore, registers that are not clocked by any clocks will be shown using this option.		
<b>Example Usage</b>	<pre># Report the clock network that feeds into register regA report_clock_network -panel {Report} -target [get_registers regA]  # Report the clock network starting from clk_100 clock, but show only the first 10 levels report_clock_network -panel {Report} -target [get_clocks clk_100] -initial_depth 10  # Report the clock network passing through the combinational node clk_mux combout. # In the report, include paths that end at a register's clock pin but do not start # at a clock source. report_clock_network -panel {Report} -target [get_nodes clk_mux combout] - include_non_clock_paths</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.48. report\_clock\_transfers (::quartus::sta)

The following table displays information for the `report_clock_transfers` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>report_clock_transfers [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-hold] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-setup] [-stdout]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension	
	<code>-hold</code>	Creates a clock transfer summary for hold analysis	
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel	
	<code>-recovery</code>	Creates a clock transfer summary for recovery analysis	
	<code>-removal</code>	Creates a clock transfer summary for removal analysis	
	<code>-setup</code>	Creates a clock transfer summary for setup analysis	
	<code>-stdout</code>	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Generates a timing report table showing all clock transfers (i.e., data paths between one clock domain and another clock domain). The from and to clocks are shown as well as the number of paths for each transfer: RR, RF, FR, FF. An RF transfer, for example, occurs when the source register of path is clocked by the rising edge of its clock and the destination register is clocked by the falling edge of its clock. The report also indicates what clock transfers are cut ("false paths") by <code>set_clock_groups</code> or <code>clock-to-clock set_false_path</code> commands. For transfers that are not cut, the number of paths reported does not take into account paths cut by path-specific <code>set_false_path</code> commands. Actual path counts may be lower than reported. The report can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical user interface ("panel_name"), or any combination of the three. The <code>-setup</code> , <code>-hold</code> , <code>-recovery</code> , and <code>-removal</code> options determine the analysis type of the report, particularly the reporting of false_paths that apply to only one analysis type. If you do not specify any of these options, a report is generated for each analysis.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist -skip_dat report_clock_transfers -panel_name delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.
	TCL_ERROR	1	ERROR: Netlist must be updated. Run <code>update_timing_netlist</code>

### 3.1.29.49. report\_clocks (::quartus::sta)

The following table displays information for the `report_clocks` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>report_clocks [-h   -help] [-long_help] [-append] [-desc] [-file &lt;name&gt;] [-hierarchy] [-panel_name &lt;name&gt;] [-stdout] [-summary] [-tree] [-waveform]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-desc	Sort the clocks by name in descending order (ascending order is default)	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-hierarchy	Display a tree view of the clocks	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-summary	Create a single table with a summary of each clock	
	-tree	alias for hierarchy	
	-waveform	Display the clocks graphically as waveforms	
<b>Description</b>	Report can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical interface ("panel_name"), or any combination of the three. For stdout/file output, the clock details are reported in two sections. The first section shows all clocks, their period, and their waveform. This includes generated clocks after an update_timing_netlist. The second section shows details for all generated clocks. For the panel report, both sections are combined into a single report.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  report_clocks  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.50. report\_datasheet (::quartus::sta)

The following table displays information for the `report_datasheet` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>report_datasheet [-h   -help] [-long_help] [-append] [-expand_bus] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
	<i>continued...</i>		

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	<code>-expand_bus</code>	If set, bus is reported as individual ports	
	<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension	
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel	
	<code>-stdout</code>	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	<b>Description</b>	This function creates a datasheet report which summarizes the timing characteristics of the design as a whole. It reports setup (tsu), hold (th), clock-to-output (tco), minimum clock-to-output (mintco), output enable (tzx), minimum output enable (mintzx), output disable (txz), minimum output disable (mintxz), propagation delay (tpd), and minimum propagation delay (mintpd) times. These delays are reported for each clock or port for which they are relevant. If there is a case where there are multiple paths for a clock (for example if there are multiplexed clocks), then the maximum delay is reported for the tsu, th, tco, tzx, txz and tpd, and the minimum delay is reported for mintco, mintzx, mintxz and mintpd. The datasheet can be outputted to the Tcl console ("stdout", default), a file ("file"), or a report panel ("panel_name"). Additionally if the "file" option is used then the "-append" option can be used to specify that new data should be written to the end of the specified file.	
<b>Example Usage</b>	<pre>project_open proj1 create_timing_netlist read_sdc update_timing_netlist  # Report the datasheet to a report panel report_datasheet -panel_name Datasheet  # Report the datasheet to a file report_datasheet -file file1.txt</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.51. report\_ddr (::quartus::sta)

The following table displays information for the `report_ddr` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<code>report_ddr [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.

*continued...*

	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	This command generates custom timing reports for EMIF instantiations.		
<b>Example Usage</b>	<pre>project_new test create_timing_netlist read_sdc update_timing_netlist  report_ddr -panel "Report DDR"  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.52. report\_design\_metrics (::quartus::sta)

The following table displays information for the `report_design_metrics` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_design_metrics [-h   -help] [-long_help] [-append] [-detail &lt;histogram path&gt;] [-enable_complementary] [-extra_info &lt;basic all none&gt;] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack_limit&gt;] [-logic_depth] [-neighbor_path_num &lt;number&gt;] [-neighbor_paths] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;] [-topology]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-detail <histogram path>	Option to determine how much detail should be shown in the report
	-enable_complementary	Option to enable complementary analysis for the neighbor paths report
	-extra_info <basic all none>	Option to determine how much detail should be shown in the Extra Info report
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	<i>continued...</i>	

<code>-fall_from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-fall_through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)
<code>-fall_to &lt;names&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
<code>-fall_to_clock &lt;names&gt;</code>	Valid destination clocks (string patterns are matched using Tcl string matching)
<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension
<code>-from &lt;names&gt;</code>	Valid sources (string patterns are matched using Tcl string matching)
<code>-from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-hold</code>	Option to report clock hold paths
<code>-intra_clock</code>	Only report paths whose launch and latch clock are the same
<code>-less_than_slack &lt;slack limit&gt;</code>	Limit the paths reported to those with slack values less than the specified limit.
<code>-logic_depth</code>	Run the logic depth analysis
<code>-neighbor_path_num &lt;number&gt;</code>	Specifies the number of before and after paths for the neighbor paths report (default=1)
<code>-neighbor_paths</code>	Run the neighbor paths analysis
<code>-npaths &lt;number&gt;</code>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
<code>-nworst &lt;number&gt;</code>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
<code>-pairs_only</code>	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel
<code>-recovery</code>	Option to report recovery paths
<code>-removal</code>	Option to report removal paths
<code>-rise_from &lt;names&gt;</code>	Valid sources (string patterns are matched using Tcl string matching)
<code>-rise_from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-rise_through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)
<code>-rise_to &lt;names&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
<code>-rise_to_clock &lt;names&gt;</code>	Valid destination clocks (string patterns are matched using Tcl string matching)

*continued...*

	-setup	Option to report clock setup paths	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)	
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
	-topology	Topology Analysis (using arbitrary path analysis)	
<b>Description</b>	<p>Reports design metrics of the worst-case paths, with support for two report types specified by the following arguments: -logic_depth (See the help for report_logic_depth for details) -neighbor_paths (See the help for report_neighbor_paths for details) Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. In -logic_depth mode, "-topology" analysis can also be performed. The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "rise_to" and "fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "rise_through" and "fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, only the single worst-case path is provided. Use the "less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. In -logic_depth mode, Use the "detail" option to specify the desired level of report detail. Specifying "histogram" generates a summary showing the distribution of logic depth among the critical paths. A row will be provided for each clock and a column for each logic depth. Specifying "path" generates a table of paths as rows with a column corresponding the paths' logic depths similar to the tables reported by "report_timing" and "report_path". The default behavior is to report a histogram.</p>		
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Report logic depth among critical paths between # "foo" and "bar". report_design_metrics -logic_depth -from foo -to bar  # Report neighbor paths among critical paths between # clocks "clk1" and "clk2" report_design_metrics -neighbor_paths -from_clock clk1 \                      -to_clock clk2  # The following command is optional delete_timing_netlist  project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.53. report\_exceptions (::quartus::sta)

The following table displays information for the report\_exceptions Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_exceptions [-h   -help] [-long_help] [-append] [-clock_groups] [-detail &lt;summary path_summary path_only path_and_clock full_path&gt;] [-fall_from_clock &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-false_path] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-ignored] [-intra_clock] [-less_than_slack &lt;slack limit&gt;] [-max_delay] [-min_delay] [-multicycle_path] [-npaths &lt;number&gt;] [-num_exceptions &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-reachability] [-recovery] [-removal] [-report_clock_groups] [-rise_from_clock &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-split_by_corner] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;] [-valid]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-clock_groups	Option to show clock groups in reports. This option also treats clock groups as timing exceptions
	-detail <summary path_summary path_only path_and_clock full_path>	Option to determine how much detail should be shown in the path report
	-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-false_path	Option to report false path exceptions
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-hold	Option to report clock hold paths
	-ignored	Option to report only exceptions that are partially or fully ignored
	-intra_clock	Only report paths whose launch and latch clock are the same
	-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
	-max_delay	Option to report maximum delay exceptions
	-min_delay	Option to report minimum delay exceptions
	-multicycle_path	Option to report multicycle path exceptions
	-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)

*continued...*

-num_exceptions <number>	Option to only show a certain number of exceptions in the report
-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
-reachability	Option to report a percent value of how many nodes in an exception's targets are satisfied by the exception
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-report_clock_groups	Option to treat clock groups as exceptions
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths
-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.
-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-valid	Option to report only exceptions that cover valid paths and have been successfully applied
<b>Description</b>	Reports status and timing analysis results for each timing exception in your design. A timing exception is one of: set_false_path, set_multicycle_path, set_min_delay, or set_max_delay. The status is relative to the paths covered by the -from, -to, and other options. A complete timing exception relative to the values specified for the -from and -to options may not actually be complete with respect to the full design. Complete: The exception has not been overridden and is valid. There are paths affected by this exception. Partially overridden: The exception includes some paths that have been overridden by one or more higher-precedence exceptions. Fully overridden: All paths affected by this exception have been overridden by one or more higher-precedence exceptions. Invalid: No paths are affected by this exception. This occurs when a timing exception has valid -from, -to, or -through collections, but there are no actual paths from the -from nodes to the -to nodes. Paths will not be analyzed: This exception has no paths for the given analysis type (setup/hold/recovery/removal). This occurs when a timing exception has paths, but only for analysis types other than the type used for the current report. Use the -valid option to show only exceptions that have the status of "Complete" or "Partially overridden". These exceptions cover valid paths and have been successfully applied. Use the -ignored option to show only exceptions that are partially or fully ignored. This includes exceptions with the status "Partially Overridden", "Fully Overridden", "Invalid", as well as any exceptions that have errors and were not included in the analysis. Use the -setup (default), -hold, -recovery, or -removal options to further filter the exceptions reported. The report

*continued...*

can be directed to the Tcl console using -stdout (default), a file using -file, the Timing Analyzer graphical user interface using -panel\_name, or any combination of those three options. You can limit the reporting by this command to specific start and end points, using the -from and -to options. You can further limit reporting to clocks using the -from\_clock and -to\_clock options, or to specific edges of the clock using the -rise\_from\_clock, -fall\_from\_clock, -rise\_to\_clock, and -fall\_to\_clock options. Additionally, the -through option can be used to restrict reporting to paths which go through specified pins or nets. To determine which timing exceptions override other timing exceptions, use the same -from and -to options that were used with the overridden timing exception. Use the -npaths option to limit the number of paths to report per timing exception. If you do not specify this option, only the single worst-case path per timing exception is provided. Use the -less\_than\_slack option to limit output to all paths with slack less than the specified value, up to the number specified by -npaths. Use the -nworst option to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the -npaths option. If this option is used, but -npaths is not specified, -npaths defaults to the same value specified for -nworst. Use the "-pairs\_only" option to filter the output further, restricting the results to only unique combinations of start and end points. Use the "-num\_exceptions" option to limit the report to only show a certain number of exceptions. If the limit is not set or set too high, the report may need to run for extended durations. Use the "-reachability" option to determine a percentage of how many start and endpoint pairs given by an exception's -from and -to filters are satisfied by the exception. If an exception does not have a -from target, the reachability ratio measures how many of the -to targets are satisfied by the exception, and vice versa for exceptions without a -to target. Exceptions without both -from and -to targets are not calculated for reachability. An exception that targets clocks or uses wildcards that match many nodes likely has a lower reachability than an exception that targets specific nodes. To avoid overly-broad exception constraints, you should use exceptions with a higher reachability value. Reachability is calculated differently depending on whether the exception is bus-type. A bus-type exception has both -from and -to options declared, and they target the same number of nodes. As well, all nodes in the -from option must be declared together, and all nodes in the -to option must be declared together. Otherwise, the exception is non-bus type. Bus-type reachability ratio assumes that each node in the -from target connects with one node in the -to target. Non-bus type reachability ratio is typically more pessimistic because it assumes that each node in the -from target connects with every node in the -to target. Use the -detail option to specify the desired level of report detail. Specifying "summary" generates a single table listing only the highlights of each timing exception (status and worst-case slack). Specifying "path\_summary" generates a table, per timing exception, listing only the highlights of each path. Specifying "path\_only" reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. This is the default behavior. Specifying "path\_and\_clock" extends the arrival and required paths back to the launch and latch clocks. Specifying "full\_path" causes continued tracing back through generated clocks to the underlying base clock. To treat clock groups as timing exceptions (meaning that they will override exceptions with a lower priority), use the "-report\_clock\_groups" option. By default, all exception types are reported, and clock groups are reported only if the "-report\_clock\_groups" option is used. Use a combination of "-false\_path", "-multicycle\_path", "-max\_delay", "-min\_delay", and "-clock\_groups" to limit the analysis to specific exception types. Note that "-clock\_groups" option also treats clock groups as timing exceptions. False path exceptions (set\_false\_path) are reported as if the false path was not applied, similar to the -false\_path option for report\_timing. The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use\_timing\_analyzer\_styleEscaping for details.

\*\*\*\*\* Clock-Domains-Crossing Verification: To view the effects of clock groups on your design, run "report\_exceptions -report\_clock\_groups". If any pair of clocks in your design are cut by a set\_clock\_groups command and NOT cut by a set\_false\_path command, the Report shows paths between that pair of clocks as a clock-to-clock exception. Example:

```
set_clock_groups -logically_exclusive -group {clkA clkB} -group {clkC clkD} report_exceptions -report_clock_groups -npaths 0 -detail path_only
```

You might want to run "report\_exceptions -report\_clock\_groups" if:

1. You want to check which paths are cut by a set\_clock\_groups SDC command.
2. You want to cut clock paths, but don't want to add set\_clock\_groups because you're concerned that you might cut a path incorrectly. With the "report\_exceptions -report\_clock\_groups" command, you can add set\_clock\_groups to split clocks that are truly asynchronous, then add set\_false\_path commands to manually cut what you want, and then verify (in the Report) that no paths were found (since false paths have priority).
3. If you use the set\_clock\_groups command and have some logic that is behaving badly and think it's timing related, run the "report\_exceptions -report\_clock\_groups" command on that hierarchy to verify whether the correct paths have been cut.

\*\*\*\*\*

**Example Usage**

```
# Reports all timing exceptions for a setup analysis.  
report_exceptions  
  
# Reports all timing exceptions for a hold analysis.  
report_exceptions -hold
```

*continued...*

```

# Reports all timing exceptions affecting input paths for
# recovery analysis, reporting the 10 worst paths per exception.
report_exceptions -from [all_inputs] -to [all_registers] \
    -recovery -npaths 10 -detail full_path

# Reports all paths affected by timing exceptions, including
# all clock-to-clock-paths cut by clock groups.
report_exceptions -report_clock_groups -npaths 0 -detail path_only

# Reports false path exceptions to determine which ones were overridden by clock groups
report_exceptions -false_path -report_clock_groups -npaths 20

# Reports only clock groups, multicycle paths, and min delays
report_exceptions -clock_groups -multicycle_path -min_delay

# Generate a reachability report that shows 10 exceptions
report_exceptions -reachability -num_exceptions 10

```

Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

### 3.1.29.54. report\_ini\_usage (::quartus::sta)

The following table displays information for the `report_ini_usage` Tcl command:

Tcl Package and Version	Belongs to ::quartus::sta on page 393		
Syntax	<code>report_ini_usage [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
Arguments	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
Description	Reports the list of .ini files and variables used during timing analysis.		
Example Usage	<pre> project_open top create_timing_netlist read_sdc update_timing_netlist report_ini_usage -panel "INI Usage" </pre>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.55. report\_logic\_depth (::quartus::sta)

The following table displays information for the `report_logic_depth` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<pre>report_logic_depth [-h   -help] [-long_help] [-append] [-detail &lt;histogram path&gt;] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack limit&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;] [-topology]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-detail <histogram path>	Option to determine how much detail should be shown in the report
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-hold	Option to report clock hold paths
	-intra_clock	Only report paths whose launch and latch clock are the same
<b>continued...</b>	-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
	-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
<b>continued...</b>	-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst

-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths
-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-topology	Topology Analysis (using arbitrary path analysis)
<b>Description</b>	Reports a summary showing the distribution of logic depth among the critical paths. Logic depth typically corresponds to the number of look-up tables (LUTs) that a path passes through. If a path passes through non-LUT elements such as RAM or DSP blocks, special rules may apply. Use the "-setup", "-hold", "-recovery", "-removal", or "-topology" options to specify which kind of analysis should be performed. In topology analysis, paths are ordered by logic depth as opposed to slack. The report can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical user interface ("panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "rise_to" and "fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "rise_through" and "fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, 1000 paths with the worst slack will be included in the report. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. Use the "detail" option to specify the desired level of report detail. Specifying "histogram" generates a summary showing the distribution of logic depth among the critical paths. A row will be provided for each clock and a column for each logic depth. Specifying "path" generates a table of paths as rows with a column corresponding to the paths' logic depths similar to the tables reported by "report_timing" and "report_path". The default behavior is to report a histogram. The values of the "-from", "-to", and "-through" options are either collections

*continued...*

	<p>or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details. The following options are not supported for this command: <code>--ccpp</code>, <code>--summary_view</code>, <code>--show_routing</code>, <code>--show_xtalk</code>, <code>--detail</code>, <code>--false_path</code> If any of the following filters are provided, the logic depth distribution will occur over "n critical paths" rather than "n critical paths per clock" -from, -to, -through, -from_clock, -to_clock, -rise_from, -fall_from, -rise_to, -fall_to, -rise_from_clock, -fall_from_clock, -rise_to_clock, -fall_to_clock, -rise_through, -fall_through</p>																		
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  report_logic_depth -npaths 1000 -file "logic_depth.txt"  project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Option &lt;string&gt; has illegal value: &lt;string&gt;. Specify a legal option value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Collection type '&lt;string&gt;' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.	TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open
Code Name	Code	String Return																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.																	
TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.																	
TCL_ERROR	1	ERROR: Report database is not open																	

### 3.1.29.56. `report_max_clock_skew` (::quartus::sta)

The following table displays information for the `report_max_clock_skew` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_max_clock_skew [-h   -help] [-long_help] [-append] [-detail &lt;summary full_path&gt;] [-file &lt;name&gt;] [-less_than_slack &lt;slack limit&gt;] [-npaths &lt;number&gt;] [-panel_name &lt;name&gt;] [-show_routing] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-detail <summary full_path>	Option to determine how much detail should be shown in the path report
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
	-npaths <number>	Specifies the number of paths to report for each latest and earliest arrival skew result per set_max_skew assignment (default=1)
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel

*continued...*

	-show_routing	Option to display detailed routing in the path report	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	TODO		
<b>Example Usage</b>	#TODO report_max_clock_skew		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.57. report\_max\_skew (::quartus::sta)

The following table displays information for the `report_max_skew` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_max_skew [-h   -help] [-long_help] [-append] [-detail &lt;summary path_only path_and_clock full_path&gt;] [-file &lt;name&gt;] [-less_than_slack &lt;slack limit&gt;] [-npaths &lt;number&gt;] [-panel_name &lt;name&gt;] [-show_routing] [-stdout]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-detail <summary path_only path_and_clock full_path>	Option to determine how much detail should be shown in the path report
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
	-npaths <number>	Specifies the number of paths to report for each latest and earliest arrival skew result per set_max_skew assignment (default=1)
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-show_routing	Option to display detailed routing in the path report
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
<b>Description</b>	<p>Reports max skew analysis results for all set_max_skew commands in a single report. For each valid set_max_skew constraint, this command computes skew with respect to the latest and the earliest arrival of each path. By default, "Skew for the Latest Arrival" is computed by comparing the latest arrival of each path with the earliest arrival of the path that has the smallest value for early arrival of all other paths included in the constraint. Similarly, "Skew for the Earliest Arrival" is computed by comparing the earliest arrival of each path with the latest arrival of the path that has the largest value for late arrival of all other paths included in the constraint. No path is compared with itself. Use the -stdout option to direct the report to the Tcl console (default), the -file option to write the report to a</p>	

*continued...*

	<p>file or the -panel_name option to direct the report to the Timing Analyzer graphical user interface. You can use these options in any combination. Use the -npaths option to limit the number of path result pairs reported for each set_max_skew constraint. If you do not specify this option, report_max_skew only reports the result pair for the single worst-case path. Use the -less_than_slack option to limit output to all paths with slack less than the specified value, up to the number specified with -npaths. Use the -detail option to specify the desired level of report detail. The -detail summary option generates a single table listing only the highlights of each path (and is the same as -summary option, which this replaces). "-detail path_only" (default) reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. "-detail path_and_clock" extends the arrival and required paths back to the launch and latch clocks. "-detail full_path" continues tracing back through generated clocks to the underlying base clock. The -show_routing option displays detailed routing information in the path. Lines marked "IC" without the option are shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line. The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case slack, in terms of the current default time unit. The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path. Possible "RF" values are:</p> <table border="0"> <tr> <td>Value Description -----</td><td>(empty) Unknown transition</td><td>R Rising</td><td>F Falling</td></tr> <tr> <td>output</td><td>RR</td><td>RF</td><td>FR</td></tr> <tr> <td>RR</td><td>Rising input, rising output</td><td>RF</td><td>Falling input, rising output</td></tr> <tr> <td>RF</td><td>Rising input, falling output</td><td>FR</td><td>Falling input, falling output</td></tr> </table> <p>The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path. Possible "Type" values are:</p> <table border="0"> <tr> <td>Value Description -----</td><td>CELL</td><td>COMP</td><td>PLL</td><td>clock</td></tr> <tr> <td>delay</td><td>Cell delay</td><td>Compensation delay</td><td>PLL clock</td><td>Interconnect delay</td></tr> <tr> <td>IC</td><td>Interconnect delay</td><td>iExt</td><td>External input</td><td>delay</td></tr> <tr> <td>loop</td><td>Lumped combinational loop</td><td>oExt</td><td>External output</td><td>delay</td></tr> <tr> <td>RE</td><td>Routing element (only for paths generated with the -show_routing option)</td><td>uTco</td><td>Register micro-Tco time</td><td>uTsu</td><td>Register micro-Tsu time</td></tr> <tr> <td>uTh</td><td>Register micro-Th time</td><td>uTh</td><td>Register micro-Th time</td><td>uTh</td><td>Register micro-Th time</td></tr> </table>	Value Description -----	(empty) Unknown transition	R Rising	F Falling	output	RR	RF	FR	RR	Rising input, rising output	RF	Falling input, rising output	RF	Rising input, falling output	FR	Falling input, falling output	Value Description -----	CELL	COMP	PLL	clock	delay	Cell delay	Compensation delay	PLL clock	Interconnect delay	IC	Interconnect delay	iExt	External input	delay	loop	Lumped combinational loop	oExt	External output	delay	RE	Routing element (only for paths generated with the -show_routing option)	uTco	Register micro-Tco time	uTsu	Register micro-Tsu time	uTh	Register micro-Th time	uTh	Register micro-Th time	uTh	Register micro-Th time
Value Description -----	(empty) Unknown transition	R Rising	F Falling																																														
output	RR	RF	FR																																														
RR	Rising input, rising output	RF	Falling input, rising output																																														
RF	Rising input, falling output	FR	Falling input, falling output																																														
Value Description -----	CELL	COMP	PLL	clock																																													
delay	Cell delay	Compensation delay	PLL clock	Interconnect delay																																													
IC	Interconnect delay	iExt	External input	delay																																													
loop	Lumped combinational loop	oExt	External output	delay																																													
RE	Routing element (only for paths generated with the -show_routing option)	uTco	Register micro-Tco time	uTsu	Register micro-Tsu time																																												
uTh	Register micro-Th time	uTh	Register micro-Th time	uTh	Register micro-Th time																																												
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  # create max skew constraints set_max_skew -from [get_ports data_ports[*]] -to [get_keepers *] 0.200 set_max_skew -from [get_keepers *] -to [get_ports output_ports[*]] 0.100  # show worst 10 paths for each earliest and latest arrival results # per max skew assignment assuming that their slack is less than 0.100 report_max_skew -panel_name "Report Max Skew" -npaths 10 -less_than_slack 0.100 -detail full_path  delete_timing_netlist project_close</pre>																																																
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open																																				
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>																																															
TCL_OK	0	INFO: Operation successful																																															
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.																																															
TCL_ERROR	1	ERROR: Report database is not open																																															

### 3.1.29.58. report\_metastability (::quartus::sta)

The following table displays information for the report\_metastability Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	report_metastability [-h   -help] [-long_help] [-append] [-file <name>] [-length <number>] [-max_length <number>] [-min_length <number>] [-nchains <number>] [-panel_name <name>] [-stdout]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values

*continued...*

	<p>-append</p> <p>-file &lt;name&gt;</p> <p>-length &lt;number&gt;</p> <p>-max_length &lt;number&gt;</p> <p>-min_length &lt;number&gt;</p> <p>-nchains &lt;number&gt;</p> <p>-panel_name &lt;name&gt;</p> <p>-stdout</p>	<p>If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.</p> <p>Sends the results to an ASCII or HTML file. Depending on the extension</p> <p>Reports only the synchronizer chains of an exact length</p> <p>Specifies the maximum length of a chain that appears in the report (default=no limit)</p> <p>Specifies the minimum length of a chain that appears in the report (default=0)</p> <p>Specifies the number of chains to report (default=1)</p> <p>Sends the results to the panel and specifies the name of the new panel</p> <p>Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.</p>
<b>Description</b>	<p>Report can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical interface ("panel_name"), or any combination of the three. The report_metastability function can be used to estimate the robustness of asynchronous transfers in your design. ----- Background ----- Synchronization register chains should be used when transferring data between unrelated clock domains to greatly reduce the probability of the captured data signal becoming metastable. A synchronization register chain is a sequence of registers with the same clock, that is driven by a pin, or logic from an unrelated clock domain. The output of all but the last register in the chain must connect only to the next register, either directly or indirectly through logic. When a register is metastable, its output hovers at a voltage between high and low for a length of time beyond the normal TCO for the register. The design can fail if subsequent registers that use this metastable signal latch different values. Therefore, it is important to properly synchronize data signals to prevent such occurrences. ----- Output ----- The report_metastability function generates a list of synchronization register chains found in the design, and can provide estimates of the Mean Time Between Failures (MTBF) of each chain. The design MTBF is an estimate of the overall robustness of the design, computed from the MTBF results from all synchronization chains with calculated MTBFs. The design MTBF metric is reported only when the design meets timing. Therefore, it is important to fully timing constrain your design. The typical MTBF result assumes typical silicon characteristics for the selected device speed grade, with nominal operating conditions. The worst case MTBF result uses the worst case silicon characteristics for the selected device speed grade, with worst case operating conditions. ----- Settings ----- To get a list of possible synchronization chains, set "Synchronizer Identification" to AUTO in the Timing Analyzer Page in the Settings dialog box. This will set the "Synchronizer Identification" QSF assignment in your QSF file. The Timing Analyzer will use timing constraints to automatically detect synchronization chains in the design. Metastability analysis checks for signal transfers between circuitry in unrelated or asynchronous clock domains, so clock domains must be related correctly with the timing constraints. Set the maximum number of registers to consider as part of one synchronization chain, via the "Synchronization Register Chain Length" setting under Analysis and Synthesis Page in the Settings dialog box. The default length is 2. All the registers in a chain (up to this length) will be protected from optimizations that can decrease MTBF. Note that if you change the "Synchronizer Identification" setting, you should rerun the Fitter, as this setting can impact some optimization algorithms. Use the -nchains option to limit the number of chains to report. If you do not specify this option, only the single worst-case chain is reported. To filter the synchronizer chains by their lengths, use the -min_length option to set the minimum chain length to be reported, and use the -max_length option to set the maximum chain length to be reported. Alternatively, use the -length option to report only chains with a specific length. ----- Report Panels ----- The MTBF Summary report provides the estimated mean time between failure for the design. This is an estimate for the overall robustness of the design in terms of metastability, and it is computed from all available synchronization chain MTBFs present in the design. The MTBF metric of automatically identified synchronization chains is not computed. To compute the MTBF of a synchronization chain, set "Synchronizer Identification" to "Forced If Asynchronous" or "Forced" for all registers of the synchronization chain. By explicitly specifying that this synchronization chain is valid, this chain will then be optimized during the Fitter, and its MTBF will be computed. Its MTBF will then be included in the computation of the design MTBF. The Synchronizer Summary table lists all the synchronization chains found in your design. It is possible that the analysis performed might erroneously interpret certain structures, such as shift registers, as synchronization chains. If some synchronization chains are misidentified and you wish to remove them from the report, you can turn off analysis of these paths by making node-based assignments via the Assignment Editor, set</p>	

*continued...*

	<p>"Synchronizer Identification" to "Off" for the first register in these synchronization chains. Conversely, if there are synchronization chains in your design that were not detected, you can set "Synchronizer Identification" assignment to "Forced If Asynchronous" for all registers in this chain through the Assignment Editor, and this chain will be reported if it meets the criteria for being a synchronization chain. This can often occur if there is logic present between the registers of the synchronization chain. In the automatic mode of synchronizer identification, these structures are not considered to be synchronizers. If you want to force a register to be identified as the head of a synchronizer, set the "Synchronizer Identification" assignment to "Forced" to the register, and it will always be identified as the first of a synchronization chain. This setting should not be applied to the entire design, since this will identify every register in the design as a synchronizer. The MTBF estimates assume the data being synchronized is switching at a toggle rate of 12.5% of the source clock frequency. That is, the estimates assume that the arriving data signal switches once every 8 source clock cycles. If multiple clocks apply, the highest frequency is used. If no source clocks can be determined, then the data rate is taken as 12.5% of the synchronization clock frequency. If you know the approximate rate at which the data changes, and would like to obtain a more accurate MTBF, use the "Synchronizer Toggle Rate" assignment in the Assignment Editor. Set the data toggle rate, in number of transitions per second, on the first register of a synchronization chain. The Timing Analyzer will then take the specified rate into account when computing the MTBF of that particular chain. You can also apply this assignment to an entity or the entire design. Since a "Synchronizer Toggle Rate" assignment of 0 indicates that the data signal never toggles, the affected synchronization chain will not be reported since it does not affect the reliability of the design.</p>						
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  report_metastability  # Reports only 10 chains that are between 2 and 4 registers long report_metastability -min_length 2 -max_length 4 -nchains 10  delete_timing_netlist project_close</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>					
TCL_OK	0	INFO: Operation successful					

### 3.1.29.59. report\_min\_pulse\_width (::quartus::sta)

The following table displays information for the report\_min\_pulse\_width Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_min_pulse_width [-h   -help] [-long_help] [-append] [-detail &lt;summary full_path&gt;] [-file &lt;name&gt;] [-nworst &lt;number&gt;] [-panel_name &lt;name&gt;] [-show_routing] [-split_by_corner] [-stdout] [-type &lt;all min_period clock_pulse&gt;] [&lt;targets&gt;]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-detail <summary full_path>	Option to determine how much detail should be shown in the report
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-nworst <number>	Specifies the number of pulse width checks to report (default=1)
	<i>continued...</i>	

	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-show_routing	Option to display detailed routing in the path report	
	-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-type <all min_period clock_pulse>	Option to determine the minimum pulse width analysis type	
	<targets>	Either clocks or nodes such as ports and registers.	
<b>Description</b>	Reports the results of minimum pulse width and minimum period checks. A minimum pulse width check verifies that a clock high ("High") or low ("Low") pulse sustains long enough to qualify as a recognizable change in the clock signal at a register clock pin. A failed minimum pulse width check indicates that the register may not recognize the clock transition. Each register in the design is reported twice per clock for minimum pulse width checks: once for the high pulse and once for the low pulse. A minimum period check verifies that the clock period ("Period") is large enough for the device to operate. Minimum period checks apply to many types of resources in the FPGA device. The results of the minimum pulse width checks can be output to the Tcl console ("stdout," the default), a report panel ("panel"), a file ("file"), or a combination of the three. Results are sorted from worst-case slack to best-case slack. To limit the number of checks reported, use the "-nworst" option. Results can be shown in summary ("detail summary") or in detail ("detail full_path," the default), showing the path details for the clock arrival times and how they affect the actual pulse width. The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  # Report the worst 100 minimum pulse width checks report_min_pulse_width -nworst 100  # Report minimum pulse width checks for the register test_reg[*] report_min_pulse_width test_reg[*]  # Output the previous results to a report panel and a file. report_min_pulse_width -panel_name "Min Pulse (test_reg)" test_reg[*]  # Output the previous results to a file. report_min_pulse_width -file min_pulse_test_reg.txt test_reg[*]</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.60. report\_neighbor\_paths (::quartus::sta)

The following table displays information for the report\_neighbor\_paths Tcl command:

Tcl Package and Version	Belongs to ::quartus::sta on page 393
Syntax	report_neighbor_paths [-h   -help] [-long_help] [-append] [-enable_complementary] [-extra_info <basic all none>] [-fall_from <names>] [-fall_from_clock <names>] [-fall_through <names>] [-fall_to <names>] [-

*continued...*

	<pre>fall_to_clock &lt;names&gt; ] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack limit&gt;] [-neighbor_path_num &lt;number&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-enable_complementary	Option to enable complementary analysis for the neighbor paths report
	-extra_info <basic all none>	Option to determine how much detail should be shown in the Extra Info report
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-hold	Option to report clock hold paths
	-intra_clock	Only report paths whose launch and latch clock are the same
	-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
	-neighbor_path_num <number>	Specifies the number of before and after paths for the neighbor paths report (default=1)
	-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
	-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
	-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.

*continued...*

-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths
-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
<b>Description</b>	Reports the most timing-critical paths in the design, including associated slack and additional path summary information, including path bounding boxes. Additionally, for each path, shows its most timing-critical neighbor paths: the path with the worst slack that fans into the source of the current path (shown on the report as "Critical Path Before") and the path with the worst slack that fans out of the destination of the current path (shown as "Critical Path After"). Note that if a register's output is connected to its own input, one or both of the neighbor paths may be identical to the current path. When looking for neighbor paths with the worst slack, all operating conditions are considered, not just the operating conditions of the main path itself. Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "rise_to" and "fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "rise_through" and "fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, only the single worst-case path is provided. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details. The following options are not supported for this command: --ccpp, --summary_view, --show_routing, --show_xtalk, --false_path Ideally you should use -panel_name or -file option to get the best view of the report. Here is description of each of the rows displayed for each path: From: Source node in the path To: Destination node in the path Launch Clock: Source node clock Latch Clock: Destination node clock Relationship: Clock relationship Slack:

**continued...**

	<p>Slack on the path Complement Analysis Slack: Slack of the same path via complement analysis Number of Paths: Number of paths with the same source and destination nodes Clock Skew: Clock Skew on the path Data Delay: Data delay on the path ( ~ cell delay + interconnect delay + misc. delay ) uTCO Delay: input delay Cell Delay: Cell delay Interconnect Delay: Wire Delay Misc. Delay: Other kinds of delay Uncertainty Delay: Clock uncertainty delay uTSU Delay: Data stability delay - setup uTH Delay: Data stability delay - hold Logic Levels: Logic levels in the path Max Fanout: Maximum fanout for any node on the path I/O Crossings: Number of I/O crossings in the path Number of wires: number of wires encountered on the path Source/Destination Bounding Box: Co-ordinates covered by source destination blocks Cell Bounding Box: Co-ordinates covered by cell blocks Interconnect Bounding Box: Co-ordinates covered by wires Source/Destination Relative Area: Normalized to 1.0 Cell Relative Area: Ratio of cell area against src/dst bounding area Interconnect Relative Area: Ration of wire area against src/dst bounding area Elements on Path: Displays the element type that make up that path TDB Names Along Path: All the TDB elements that make up the path (only available with developer license) Corner: Describes the corner the path is found in Complement Analysis Corner: Corner of the complement analysis path found</p>																		
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  report_neighbor_paths -npaths 10 -file "neighbor_path_analysis.txt"  # The following command is optional delete_timing_netlist  project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Option &lt;string&gt; has illegal value: &lt;string&gt;. Specify a legal option value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Collection type '&lt;string&gt;' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.	TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.																	
TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.																	
TCL_ERROR	1	ERROR: Report database is not open																	

### 3.1.29.61. report\_net\_delay (::quartus::sta)

The following table displays information for the `report_net_delay` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_net_delay [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-nworst &lt;number&gt;] [-panel_name &lt;name&gt;] [-split_by_corner] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-nworst <number>	Specifies the maximum number of paths to report for each analysis. If unspecified, there is no limit.
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	<b>continued...</b>	

	-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Reports net delay analysis results based on set_net_delay commands. Each set_net_delay command is treated as a separate analysis and report_net_delay reports the results of all set_net_delay commands in a single report. The report contains each set_net_delay command with the worst case slack result followed by the results of each edge matching the criteria set by that set_net_delay command. These results are ordered based on the slack value. Use -nworst option to limit the number of lines reported for a set_net_delay command.		
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  set_net_delay -min 0.160 -from [get_pins inst9 combout] -to [get_pins * dataaf] set_net_delay -max 0.500 -from inst8 combout  report_net_delay -panel "Net Delay"</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.62. report\_net\_timing (::quartus::sta)

The following table displays information for the report\_net\_timing Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	report_net_timing [-h   -help] [-long_help] [-append] [-file <name>] [-nworst_delay <number>] [-nworst_fanout <number>] [-panel_name <name>] [-stdout] [ <name> ]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-nworst_delay <number>	Report worst net delays
	-nworst_fanout <number>	Report worst fanout nets
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
	<name>	Signal or collection name

*continued...*

<b>Description</b>	Reports delay and fanout information about a net in the design. A net corresponds to a cell output pin. Report can be directed to the Tcl console (""-stdout", default), a file (""-file"), the Timing Analyzer graphical interface (""-panel_name"), or any combination of the three. The value of the name is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.																		
<b>Example Usage</b>	<pre>project_open &lt;design&gt; create_timing_netlist  # Show delay and fanout information for all nets # that match "abc*" report_net_timing [get_nets abc*]  # Report delay and fanout information for the 10 # nets showing higher delays report_net_timing -nworst_delay 10  # Report delay and fanout information for the 10 # nets showing higher fanout report_net_timing -nworst_fanout 10  project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Options <code>-&lt;string&gt;</code> and <code>-&lt;string&gt;</code> are mutually exclusive. Specify only one of the two options.</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: You must open a project before you can use this command.</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Neither of options <code>-&lt;string&gt;</code> or <code>-&lt;string&gt;</code> is specified. Specify one of the two options.</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Options <code>-&lt;string&gt;</code> and <code>-&lt;string&gt;</code> are mutually exclusive. Specify only one of the two options.	TCL_ERROR	1	ERROR: You must open a project before you can use this command.	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.	TCL_ERROR	1	ERROR: Neither of options <code>-&lt;string&gt;</code> or <code>-&lt;string&gt;</code> is specified. Specify one of the two options.
Code Name	Code	String Return																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Options <code>-&lt;string&gt;</code> and <code>-&lt;string&gt;</code> are mutually exclusive. Specify only one of the two options.																	
TCL_ERROR	1	ERROR: You must open a project before you can use this command.																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.																	
TCL_ERROR	1	ERROR: Neither of options <code>-&lt;string&gt;</code> or <code>-&lt;string&gt;</code> is specified. Specify one of the two options.																	

### 3.1.29.63. report\_partitions (::quartus::sta)

The following table displays information for the `report_partitions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<code>report_partitions [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-from_clock &lt;names&gt;] [-nworst &lt;number&gt;] [-panel_name &lt;name&gt;] [-stdout] [-to_clock &lt;names&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension
	<code>-from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
	<code>-nworst &lt;number&gt;</code>	Specifies the maximum number of paths to report between partitions. If unspecified, the limit defaults to 1000
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel

*continued...*

	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
<b>Description</b>	Reports timing information related to design partitions. The report_partitions command analyzes the worst 1000 failing setup paths in the design by default, but you can optionally set the nworst option to increase or decrease this number. The from_clock and to_clock arguments can be used to control how this function finds the paths to be analyzed. This function reports the number of failing paths within each partition and the worst-case slack of the paths that are entirely contained within a single partition in a Partition Timing Overview table. The function also creates a Partition Timing Details table that lists the number of failing paths and worst-case slack of paths that feed from one partition to another partition. This information provides more details on where the critical paths in the design are with respect to design partitions. The function also creates a Partition Timing Breakdown table. This table bins the worst failing paths by all partitions that the path spans, and reports the number of failing paths and worst case slack for each group of partitions. This report can be directed to the Tcl console ("stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("panel_name"), or any combination of the three.		
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  # Report a maximum of 500 failing paths between partitions to the # Timing Analyzer graphical interface and to the Tcl console. report_partitions -panel_name "Partition Timing Report" -nworst 500 -stdout</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Design partitions are not supported in this project.

### 3.1.29.64. report\_path (::quartus::sta)

The following table displays information for the report\_path Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_path [-h   -help] [-long_help] [-append] [-fall_from &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-file &lt;name&gt;] [-from &lt;names&gt;] [-list_clocks] [-logic_depth] [-min_path] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-rise_from &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-show_routing] [-split_by_corner] [-stdout] [-summary] [-through &lt;names&gt;] [-to &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)
	-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
	-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension

*continued...*

-from <names>	Valid sources (string patterns are matched using Tcl string matching)
-list_clocks	Includes the driving clocks associated with the from and to nodes of each path in the Path Summary table
-logic_depth	Option to display the logic depth instead of path delay
-min_path	Find the minimum delay path(s)
-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same as nworst
-pairs_only	When set, paths with the same start and end points will be considered to be equivalent. Only the longest delay path for each unique combination will be displayed.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-show_routing	Option to display detailed routing in the path
-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.
-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
-summary	Create a single table with a summary of each path found
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
<b>Description</b>	Reports the longest delay paths and the corresponding delay value. The report can be directed to the Tcl console ("stdout", default), a file ("file"), the Timing Analyzer graphical user interface ("panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Any node or cell in the design is considered a valid endpoint. Additionally, the "through" option can be used to restrict analysis to paths which go through specified pins or nets. Paths that are reported can not start before or go beyond a keeper node (register or port); this restriction considers register pins as combinational nodes in the design. Use "-npaths" to limit the number of paths to report. If this option is not specified, only the single longest delay path is provided. Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. Use the "-summary" option to generate a single table listing only the highlights of each path. The "-min_path" option will find the minimum delay path(s) rather than the maximum delay paths which is the default behavior. The "-show_routing" option will display detailed routing

*continued...*

	<p>information in the path. Lines that were marked as "IC" without the option will still be shown, but only as a placeholder. The routing elements for that line will be broken out individually and listed before the line. The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the longest delay, in terms of the current default time unit. The values of the "-from", "-to", "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.</p>																		
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Report path delay between nodes "foo" and "bar", # reporting the longest delay if a path is found.  set my_list [report_path -from foo -to bar] set num_paths [lindex \$my_list 0] set longest_delay [lindex \$my_list 1] if { \$num_paths &gt; 0 } {     puts "Longest delay -from foo -to bar is \$longest_delay" }  # The following command is optional delete_timing_netlist  project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Option &lt;string&gt; has illegal value: &lt;string&gt;. Specify a legal option value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Collection type '&lt;string&gt;' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.	TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.																	
TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use <code>create_timing_netlist</code> to create a timing netlist.																	
TCL_ERROR	1	ERROR: Report database is not open																	

### 3.1.29.65. report\_pipelining\_info (::quartus::sta)

The following table displays information for the `report_pipelining_info` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_pipelining_info [-h   -help] [-long_help] [-append] [-bus_name &lt;names&gt;] [-file &lt;name&gt;] [-max_rows &lt;number&gt;] [-min_depth &lt;number&gt;] [-min_width &lt;number&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>	
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> <li>-long_help</li> <li>-append</li> <li>-bus_name &lt;names&gt;</li> <li>-file &lt;name&gt;</li> </ul>	<ul style="list-style-type: none"> <li>Short help</li> <li>Long help with examples and possible return values</li> <li>If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.</li> <li>When set, pipelining report will give detailed information on the specific bus</li> <li>Sends the results to an ASCII or HTML file. Depending on the extension</li> </ul>

*continued...*

	-max_rows <number>	Specifies the maximum number of rows to report	
	-min_depth <number>	Specifies the minimum average bus depth	
	-min_width <number>	Specifies the minimum bus width	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	This command reports back-to-back pipelining registers information and collapses pipelining chains based on which bus they are in. Use the "-min_depth" option to specify the minimum average depths of pipelining buses to report. The default value for this field is "3". Use the "-min_width" option to specify the minimum width of pipelining buses to report. The default value for this field is "16". By default, the command reports all of the pipelining buses with average depth and width over the value specified by "-min_depth" and "-min_width" options. To help check one pipelining bus in detail, this command also provides detailed mode. Use the "bus_name" to specify which bus to focus.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.66. report\_register\_spread (::quartus::sta)

The following table displays information for the `report_register_spread` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_register_spread [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-from_clock &lt;names&gt;] [-num_registers &lt;number&gt;] [-panel_name &lt;name&gt;] [-sink_type &lt;endpoint immediate&gt;] [-spread_type &lt;tension span count&gt;] [-stdout]</code>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
	-num_registers <number>	Specifies the top N registers with highest spread
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-sink_type <endpoint immediate>	Determines which sink type [endpoint immediate] is analyzed
	-spread_type <tension span count>	Determines which spread type is analyzed [tension span count]
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

*continued...*

<b>Description</b>	This report analyzes the final placement of a design and strive to identify registers with sinks that are pulling them in various directions. These registers are then recommended as candidates for duplication. There are two types of sink: "Immediate Fan-Out" and "Timing Path Endpoint". There are two types of pull: "Tension" and "Span". The report can be directed to the Tcl console (""-stdout", default), a file (""-file"), the Timing Analyzer graphical user interface (""-panel_name"), or any combination of the three. Use "-sink_type" to specify between endpoints and immediate fanouts to report on. If not specified the default will report on endpoint fanouts. Timing Path Endpoints: the nodes (usually registers) that terminate timing paths from a register. Immediate Fanouts: the immediately connected nodes (lookup tables, other registers, RAM or DSP blocks, etc.) of the register. Use "-spread_type" to specify the type of spread to report on, a user can choose between: Tension: the sum over each sink of the distance from it to the centroid of all the sinks. Span: the maximum 1-dimensional delta between the left/bottom-most sink and the right/top-most sink. Count: the number of each sink type associated with the source register. The analysis can be limited to clocks using the "-from_clock" option. Use "-num_registers" to limit the number source registers reported. Registers are reported in decreasing order of the spread type selected so the top n registers will be reported. If you do not specify this option, the number of source registers displayed is limited to a maximum of 10.						
<b>Example Usage</b>	<pre>project_open my_project # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  report_register_spread -num_registers 20 -spread_type "tension" -sink_type "endpoint" project_close</pre>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful
Code Name	Code	String Return					
TCL_OK	0	INFO: Operation successful					

### 3.1.29.67. report\_reset\_statistics (::quartus::sta)

The following table displays information for the `report_reset_statistics` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393								
<b>Syntax</b>	<code>report_reset_statistics [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>								
<b>Arguments</b>	-h   -help	Short help							
	-long_help	Long help with examples and possible return values							
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.							
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension							
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel							
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.							
<b>Description</b>	This command reports reset statistics.								
<b>Example Usage</b>	This command currently contains no example usage.								
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful		
Code Name	Code	String Return							
TCL_OK	0	INFO: Operation successful							

### 3.1.29.68. report\_retimining\_restrictions (::quartus::sta)

The following table displays information for the `report_retimining_restrictions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>report_retimining_restrictions [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	This command reports retiming restrictions in a hierarchical form. The report helps users identify various types of retiming restrictions in each entity. Consider removing these retiming restrictions to allow retiming optimization to improve performance.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.69. report\_route\_net\_of\_interest (::quartus::sta)

The following table displays information for the `report_route_net_of_interest` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>report_route_net_of_interest [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-num_nets &lt;number&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-num_nets <number>	Specifies the number of nets of interest to report	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	

*continued...*

	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.			
<b>Description</b>	This command reports nets that router works hardest on.				
<b>Example Usage</b>	This command currently contains no example usage.				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

### 3.1.29.70. report\_rskm (::quartus::sta)

The following table displays information for the `report_rskm` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	<code>report_rskm [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-panel_name &lt;name&gt;] [-stdout]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Reports RSKM for dedicated LVDS circuitry. In designs that use dedicated LVDS circuitry, receiver input skew margin (RSKM) is the time margin available before the LVDS receiver megafunction fails to operate. RSKM is defined as the total time margin that remains after subtracting the sampling window (SW) size and the receiver channel-to-channel skew (RCCS) from the time unit interval (TUI), as expressed in the following formula: $RSKM = (TUI - SW - RCCS) / 2$ . The time unit interval is the LVDS clock period ( $1/fmax$ ). The sampling window is the period of time that the input data must be stable to ensure that the data is successfully sampled by the LVDS receiver megafunction. The sampling window size varies by device speed grade. RCCS is the difference between the fastest and slowest data output transitions, including the tco variation and clock skew. To obtain an accurate analysis of an LVDS circuit, you should assign an appropriate input delay to the LVDS receiver megafunction. RCCS is equal to the difference between maximum input delay and minimum input delay. If no input delay is set, RCCS defaults to zero.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  # Ensure a tccs of 1ns set_input_delay -max -clock lvds_clk 2ns [get_ports lvds_input] set_input_delay -min -clock lvds_clk 1ns [get_ports lvds_input]  # Show lvds information report_rskm</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>

*continued...*

TCL_OK	0	INFO: Operation successful
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.71. report\_sdc (::quartus::sta)

The following table displays information for the `report_sdc` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393		
<b>Syntax</b>	<code>report_sdc [-h   -help] [-long_help] [-append] [-file &lt;name&gt;] [-ignored] [-panel_name &lt;name&gt;] [-stdout]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension	
	-ignored	Reports full history of assignments to locate ignored ones	
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel	
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Reports all SDC constraints used in the design. Use the -ignored option to report SDC constraints that were ignored and the reason they were ignored.		
<b>Example Usage</b>	<pre>project_new test create_timing_netlist create_clock -period 10 -name clk10 clk set_multicycle_path -from [get_cells a] -to [get_cells b] update_timing_netlist  report_sdc -panel_name sdc_report_panel report_timing delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.72. report\_skew (::quartus::sta)

The following table displays information for the report\_skew Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393
<b>Syntax</b>	<code>report_skew [-h   -help] [-long_help] [-append] [-detail &lt;summary path_only path_and_clock full_path&gt;] [-fall_from_clock &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-greater_than_skew &lt;slack_limit&gt;] [-intra_clock] [-npaths &lt;number&gt;] [-panel_name &lt;name&gt;] [-rise_from_clock &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-show_routing] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</code>
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help Short help</li> <li>-long_help Long help with examples and possible return values</li> <li>-append If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.</li> <li>-detail &lt;summary path_only path_and_clock full_path&gt; Option to determine how much detail should be shown in the path report</li> <li>-fall_from_clock &lt;names&gt; Valid source clocks (string patterns are matched using Tcl string matching)</li> <li>-fall_to_clock &lt;names&gt; Valid destination clocks (string patterns are matched using Tcl string matching)</li> <li>-file &lt;name&gt; Sends the results to an ASCII or HTML file. Depending on the extension</li> <li>-from &lt;names&gt; Valid sources (string patterns are matched using Tcl string matching)</li> <li>-from_clock &lt;names&gt; Valid source clocks (string patterns are matched using Tcl string matching)</li> <li>-greater_than_skew &lt;slack_limit&gt; Limit the paths reported to those with skew values greater than the specified limit.</li> <li>-intra_clock Only report paths whose launch and latch clock are the same</li> <li>-npaths &lt;number&gt; Specifies the number of paths to report for each latest and earliest arrival skew result (default=1)</li> <li>-panel_name &lt;name&gt; Sends the results to the panel and specifies the name of the new panel</li> <li>-rise_from_clock &lt;names&gt; Valid source clocks (string patterns are matched using Tcl string matching)</li> <li>-rise_to_clock &lt;names&gt; Valid destination clocks (string patterns are matched using Tcl string matching)</li> <li>-show_routing Option to display detailed routing in the path report</li> <li>-stdout Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.</li> <li>-through &lt;names&gt; Valid through nodes (string patterns are matched using Tcl string matching)</li> <li>-to &lt;names&gt; Valid destinations (string patterns are matched using Tcl string matching)</li> </ul>

*continued...*

	<pre>-to_clock &lt;names&gt;</pre>	Valid destination clocks (string patterns are matched using Tcl string matching)												
<b>Description</b>	<p>This report performs skew analysis on selected paths. As opposed to report generated by report_max_skew command, this command does not depend on the existence of set_max_skew assignments. This report computes skew with respect to the latest and the earliest arrival of each selected path. By default, "Skew for the Latest Arrival" is computed by comparing the latest arrival of each path with the earliest arrival of the path that has the smallest value for early arrival of all other paths included in the constraint. Similarly, "Skew for the Earliest Arrival" is computed by comparing the earliest arrival of each path with the latest arrival of the path that has the largest value for late arrival of all other paths included in the constraint. No path is compared with itself. Use the -stdout option to direct the report to the Tcl console (default), the -file option to write the report to a file or the -panel_name option to direct the report to the Timing Analyzer graphical user interface. You can use these options in any combination. Report skew includes data arrival times, clock arrival times, register micro parameters, clock uncertainty, on-die variation and ccpp removal. Use the -npaths option to limit the number of path result pairs reported for each set_max_skew constraint. If you do not specify this option, report_skew only reports the result pair for the single worst-case path. Use the -less_than_slack option to limit output to all paths with skew greater than the specified value, up to the number specified with -npaths. Use the -detail option to specify the desired level of report detail. "-detail path_only" (default) reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. "-detail path_and_clock" extends the arrival and required paths back to the launch and latch clocks. "-detail full_path" continues tracing back through generated clocks to the underlying base clock. The "-detail summary" option is deprecated. The -show_routing option displays detailed routing information in the path. Lines marked "IC" without the option are shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line. The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case skew, in terms of the current default time unit. The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path. Possible "RF" values are: Value Description ----- (empty) Unknown transition R Rising output F Falling output RR Rising input, rising output RF Rising input, falling output FR Falling input, rising output FF Falling input, falling output The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path. Possible "Type" values are: Value Description ----- CELL Cell delay COMP PLL clock network compensation delay IC Interconnect delay iExt External input delay LOOP Lumped combinational loop delay oExt External output delay RE Routing element (only for paths generated with the -show_routing option) uTco Register micro-Tco time uTs Register micro-Tsu time uTh Register micro-Th time</p>													
<b>Example Usage</b>	<pre>project_open my_project create_timing_netlist read_sdc update_timing_netlist  # show worst 10 paths for each earliest and latest arrival results report_skew -from [get_ports input[*]] -to [get_registers *] -panel_name "Report Skew" - npaths 10 -greater_than_skew 0.100 -detail full_path  delete_timing_netlist project_close</pre>													
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th> <th>Code</th> <th>String Return</th> </tr> </thead> <tbody> <tr> <td>TCL_OK</td> <td>0</td> <td>INFO: Operation successful</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td> </tr> <tr> <td>TCL_ERROR</td> <td>1</td> <td>ERROR: Report database is not open</td> </tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open	
Code Name	Code	String Return												
TCL_OK	0	INFO: Operation successful												
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.												
TCL_ERROR	1	ERROR: Report database is not open												

### 3.1.29.73. report\_tccs (::quartus::sta)

The following table displays information for the report\_tccs Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393
<b>Syntax</b>	<code>report_tccs [-h   -help] [-long_help] [-append] [-file &lt;name&gt; ] [-panel_name &lt;name&gt; ] [-stdout]</code>
<i>continued...</i>	

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.	
	<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension	
	<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel	
	<code>-stdout</code>	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
<b>Description</b>	Reports TCCS for dedicated LVDS transmitters. In designs that implement the LVDS I/O standard, transmitter channel-to-channel skew (TCCS) is the timing difference between the fastest and slowest output transitions, including tco variations and clock skew.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  # Show lvds information report_tccs</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Report database is not open

### 3.1.29.74. report\_timing (::quartus::sta)

The following table displays information for the `report_timing` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<code>report_timing [-h   -help] [-long_help] [-append] [-data_delay] [-detail &lt;summary path_only path_and_clock full_path&gt;] [-extra_info &lt;basic all none&gt;] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-false_path] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack_limit&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-show_routing] [-split_by_corner] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-append</code>	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	<code>-data_delay</code>	Report only paths that are covered by a data delay assignment

*continued...*

<code>-detail &lt;summary path_only path_and_clock full_path&gt;</code>	Option to determine how much detail should be shown in the path report
<code>-extra_info &lt;basic all none&gt;</code>	Option to determine how much detail should be shown in the Extra Info report
<code>-fall_from &lt;names&gt;</code>	Valid sources (string patterns are matched using Tcl string matching)
<code>-fall_from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-fall_through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)
<code>-fall_to &lt;names&gt;</code>	Valid destinations (string patterns are matched using Tcl string matching)
<code>-fall_to_clock &lt;names&gt;</code>	Valid destination clocks (string patterns are matched using Tcl string matching)
<code>-false_path</code>	Report only paths that are cut by a false path assignment
<code>-file &lt;name&gt;</code>	Sends the results to an ASCII or HTML file. Depending on the extension
<code>-from &lt;names&gt;</code>	Valid sources (string patterns are matched using Tcl string matching)
<code>-from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-hold</code>	Option to report clock hold paths
<code>-intra_clock</code>	Only report paths whose launch and latch clock are the same
<code>-less_than_slack &lt;slack limit&gt;</code>	Limit the paths reported to those with slack values less than the specified limit.
<code>-npaths &lt;number&gt;</code>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
<code>-nworst &lt;number&gt;</code>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
<code>-pairs_only</code>	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
<code>-panel_name &lt;name&gt;</code>	Sends the results to the panel and specifies the name of the new panel
<code>-recovery</code>	Option to report recovery paths
<code>-removal</code>	Option to report removal paths
<code>-rise_from &lt;names&gt;</code>	Valid sources (string patterns are matched using Tcl string matching)
<code>-rise_from_clock &lt;names&gt;</code>	Valid source clocks (string patterns are matched using Tcl string matching)
<code>-rise_through &lt;names&gt;</code>	Valid through nodes (string patterns are matched using Tcl string matching)

*continued...*

-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)																																
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)																																
-setup	Option to report clock setup paths																																
-show_routing	Option to display detailed routing in the path																																
-split_by_corner	When set, running this command with the -panel option will create a folder containing versions of this report for selected multiple operating conditions. This option has no effect when used with the -stdout or -file options.																																
-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.																																
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)																																
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)																																
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)																																
<b>Description</b>	<p>Reports the worst-case paths and associated slack. Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. The report can be directed to the Tcl console ("stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "-rise_to" and "-fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "-rise_through" and "-fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, only the single worst-case path is provided. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. Use the "-detail" option to specify the desired level of report detail. "-summary" generates a single table listing only the highlights of each path (and is the same as "-summary" option, which this replaces). "-path_only" reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. This is the default behavior. "-path_and_clock" extends the arrival and required paths back to the launch and latch clocks. "-full_path" will continue tracing back through generated clocks to the underlying base clock. The "-show_routing" option displays detailed routing information in the path. Lines that were marked as "IC" without the option are still shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line. The "-false_path" option reports only those paths that are normally hidden by false_path assignments or clock to clock cuts. Like the default report, this option only reports constrained paths. The "-data_delay" option reports only those paths that are constrained by set_data_delay. Without this option, such paths are excluded from the report. The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case slack, in terms of the current default time unit. The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path. Possible "RF" values are:</p> <table style="margin-left: 20px;"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>-----</td> <td>Unknown transition</td> </tr> <tr> <td>R</td> <td>Rising output</td> </tr> <tr> <td>F</td> <td>Falling output</td> </tr> <tr> <td>RR</td> <td>RR Rising input, rising output</td> </tr> <tr> <td>FF</td> <td>FF Falling input, falling output</td> </tr> <tr> <td>FR</td> <td>FR Falling input, rising output</td> </tr> <tr> <td>RF</td> <td>RF Rising input, falling output</td> </tr> </table> <p>The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path. Possible "Type" values are:</p> <table style="margin-left: 20px;"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>----</td> <td>CELL Cell delay</td> </tr> <tr> <td>COMP</td> <td>PLL clock network compensation delay</td> </tr> <tr> <td>IC</td> <td>Interconnect delay</td> </tr> <tr> <td>iExt</td> <td>External input delay</td> </tr> <tr> <td>LOOP</td> <td>Lumped combinational loop delay</td> </tr> <tr> <td>oExt</td> <td>External output delay</td> </tr> <tr> <td>RE</td> <td>Routing element (only for paths generated with the -</td> </tr> </table>	Value	Description	-----	Unknown transition	R	Rising output	F	Falling output	RR	RR Rising input, rising output	FF	FF Falling input, falling output	FR	FR Falling input, rising output	RF	RF Rising input, falling output	Value	Description	----	CELL Cell delay	COMP	PLL clock network compensation delay	IC	Interconnect delay	iExt	External input delay	LOOP	Lumped combinational loop delay	oExt	External output delay	RE	Routing element (only for paths generated with the -
Value	Description																																
-----	Unknown transition																																
R	Rising output																																
F	Falling output																																
RR	RR Rising input, rising output																																
FF	FF Falling input, falling output																																
FR	FR Falling input, rising output																																
RF	RF Rising input, falling output																																
Value	Description																																
----	CELL Cell delay																																
COMP	PLL clock network compensation delay																																
IC	Interconnect delay																																
iExt	External input delay																																
LOOP	Lumped combinational loop delay																																
oExt	External output delay																																
RE	Routing element (only for paths generated with the -																																

*continued...*

	<p>show_routing option) uTco Register micro-Tco time uTsu Register micro-Tsu time uTh Register micro-Th time The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.</p>																		
<b>Example Usage</b>	<pre>project_open my_project  # Always create the netlist first create_timing_netlist read_sdc my_project.sdc update_timing_netlist  # Run a setup analysis between nodes "foo" and "bar", # reporting the worst-case slack if a path is found.  set my_list [report_timing -from foo -to bar] set num_paths [lindex \$my_list 0] set wc_slack [lindex \$my_list 1] if { \$num_paths &gt; 0 } {     puts "Worst case slack -from foo -to bar is \$wc_slack" }  # The following command is optional delete_timing_netlist  project_close</pre>																		
<b>Return Value</b>	<table border="1"> <thead> <tr> <th>Code Name</th><th>Code</th><th>String Return</th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Option &lt;string&gt; has illegal value: &lt;string&gt;. Specify a legal option value.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Collection type '&lt;string&gt;' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Report database is not open</td></tr> </tbody> </table>	Code Name	Code	String Return	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.	TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.	TCL_ERROR	1	ERROR: Report database is not open
Code Name	Code	String Return																	
TCL_OK	0	INFO: Operation successful																	
TCL_ERROR	1	ERROR: Option <string> has illegal value: <string>. Specify a legal option value.																	
TCL_ERROR	1	ERROR: Collection type '<string>' is not a valid type for a through collection. Valid collection types are 'pin' and 'net'																	
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.																	
TCL_ERROR	1	ERROR: Report database is not open																	

### 3.1.29.75. report\_timing\_by\_source\_files (::quartus::sta)

The following table displays information for the report\_timing\_by\_source\_files Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	<pre>report_timing_by_source_files [-h   -help] [-long_help] [-append] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-false_path] [-file &lt;name&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack_limit&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] [-panel_name &lt;name&gt;] [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-stdout] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</pre>	
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> <li>-long_help</li> <li>-append</li> <li>-fall_from &lt;names&gt;</li> </ul>	<ul style="list-style-type: none"> <li>Short help</li> <li>Long help with examples and possible return values</li> <li>If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.</li> <li>Valid sources (string patterns are matched using Tcl string matching)</li> </ul>

*continued...*

-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-false_path	Report only paths that are cut by a false path assignment
-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
-from <names>	Valid sources (string patterns are matched using Tcl string matching)
-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-hold	Option to report clock hold paths
-intra_clock	Only report paths whose launch and latch clock are the same
-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths

***continued...***

	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.	
	-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)	
	-to <names>	Valid destinations (string patterns are matched using Tcl string matching)	
	-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)	
<b>Description</b>	<p>The command groups the most timing critical paths to identify the most timing critical entities and source files. Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the Timing Analyzer graphical user interface ("-panel_name"), or any combination of the three. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "-rise_to" and "-fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "-rise_through" and "-fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, 1000 worst-case paths are provided. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. The "-false_path" option reports only those paths that are normally hidden by false_path assignments or clock to clock cuts. Like the default report, this option only reports constrained paths. The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for <code>use_timing_analyzer_styleEscaping</code> for details.</p>		
<b>Example Usage</b>	<pre>report_timing_by_source_files -setup -npaths 1000 -panel_name {Report Timing by Source Files}</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.76. report\_timing\_tree (::quartus::sta)

The following table displays information for the `report_timing_tree` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::sta</a> on page 393	
<b>Syntax</b>	<pre>report_timing_tree [-h   -help] [-long_help] [-fall_from &lt;names&gt;] [-fall_from_clock &lt;names&gt;] [-fall_through &lt;names&gt;] [-fall_to &lt;names&gt;] [-fall_to_clock &lt;names&gt;] [-from &lt;names&gt;] [-from_clock &lt;names&gt;] [-hold] [-intra_clock] [-less_than_slack &lt;slack limit&gt;] [-npaths &lt;number&gt;] [-nworst &lt;number&gt;] [-pairs_only] -panel_name &lt;name&gt; [-recovery] [-removal] [-rise_from &lt;names&gt;] [-rise_from_clock &lt;names&gt;] [-rise_through &lt;names&gt;] [-rise_to &lt;names&gt;] [-rise_to_clock &lt;names&gt;] [-setup] [-through &lt;names&gt;] [-to &lt;names&gt;] [-to_clock &lt;names&gt;]</pre>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-fall_from <names>	Valid sources (string patterns are matched using Tcl string matching)

*continued...*

-fall_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-fall_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-fall_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-fall_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-from <names>	Valid sources (string patterns are matched using Tcl string matching)
-from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-hold	Option to report clock hold paths
-intra_clock	Only report paths whose launch and latch clock are the same
-less_than_slack <slack limit>	Limit the paths reported to those with slack values less than the specified limit.
-npaths <number>	Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified. Value of 0 causes all paths to be reported but be wary that this may be slow)
-nworst <number>	Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
-pairs_only	When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
-panel_name <name>	Report panel_name
-recovery	Option to report recovery paths
-removal	Option to report removal paths
-rise_from <names>	Valid sources (string patterns are matched using Tcl string matching)
-rise_from_clock <names>	Valid source clocks (string patterns are matched using Tcl string matching)
-rise_through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-rise_to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-rise_to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)
-setup	Option to report clock setup paths
-through <names>	Valid through nodes (string patterns are matched using Tcl string matching)
-to <names>	Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>	Valid destination clocks (string patterns are matched using Tcl string matching)

*continued...*

<b>Description</b>	Reports the worst-case paths and associated slack grouped by design entity. Each entity level indicates the worst-case slack and number of paths at that level. Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. Use the "-panel_name" option to direct the report to the Timing Analyzer graphical user interface. You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Use the "-rise_from" and "-fall_from" options to limit the analysis to endpoints with established high or low starting states. Use the "rise_to" and "fall_to" options to limit the analysis to destination points with high or low ending states. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Use the "rise_through" and "fall_through" options to limit the analysis to intermediate points with high or low ending states. Use "-npaths" to limit the number of paths to report. If you do not specify this option, only the single worst-case path is provided. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths". Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst". Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or Timing Analyzer-extension substitution rules. See the help for use_timing_analyzer_styleEscaping for details.		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.77. report\_ucp (::quartus::sta)

The following table displays information for the report\_ucp Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	report_ucp [-h   -help] [-long_help] [-append] [-file <name>] [-panel_name <name>] [-stdout] [-summary]	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten. This option is not supported for HTML files.
	-file <name>	Sends the results to an ASCII or HTML file. Depending on the extension
	-panel_name <name>	Sends the results to the panel and specifies the name of the new panel
	-stdout	Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
	-summary	Generate everything except for the detailed paths panels.
<b>Description</b>	Reports unconstrained paths.	
<b>Example Usage</b>	<pre>project_open chiptrip create_timing_netlist read_sdc update_timing_netlist</pre>	

*continued...*

	<pre>report_ucp delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.78. set\_operating\_conditions (::quartus::sta)

The following table displays information for the `set_operating_conditions` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393				
<b>Syntax</b>	<code>set_operating_conditions [-h   -help] [-long_help] [-force_dat] [-grade &lt;c i m e a&gt;] [-model &lt;fast slow&gt;] [-speed &lt;speed&gt;] [-temperature &lt;value_in_C&gt;] [-voltage &lt;value_in_mV&gt;] [ &lt;list_of_operating_conditions&gt; ]</code>				
<b>Arguments</b>	<code>-h   -help</code>	Short help			
	<code>-long_help</code>	Long help with examples and possible return values			
	<code>-force_dat</code>	Option to force delay annotation (only done when selecting an unanalyzed corner)			
	<code>-grade &lt;c i m e a&gt;</code>	Option to specify temperature grade			
	<code>-model &lt;fast slow&gt;</code>	Option to specify timing model			
	<code>-speed &lt;speed&gt;</code>	Speed grade			
	<code>-temperature &lt;value_in_C&gt;</code>	Operating temperature			
	<code>-voltage &lt;value_in_mV&gt;</code>	Operating voltage			
	<code>&lt;list_of_operating_conditions&gt;</code>	list or collection of Operating conditions Tcl objects or names			
<b>Description</b>	Use this command to specify operating conditions different from the initial conditions used to create the timing netlist. When a timing model is not specified, the slow model is used. Voltage and temperature options must be used together. These two options are not available for all devices. The <code>get_available_operating_conditions</code> command returns the list of available operating conditions for your device. Use the <code>-speed</code> option to analyze the design at a different speed grade of the selected device. Use the <code>-grade</code> option to analyze the design at a different temperature grade. This option is provided to support what-if analysis and is not recommended for final sign-off analysis. By default, delay annotation is skipped if previously performed. Use <code>-force_dat</code> to rerun delay annotation.				
<b>Example Usage</b>	<pre>#do report timing for different operating conditions one by one foreach_in_collection op [get_available_operating_conditions] {     set_operating_conditions \$op     update_timing_netlist     report_timing }  #set aggregated report timing for all operating conditions when corner aggregation is enabled set_operating_conditions [get_available_operating_conditions] report_timing  #manually set operating conditions set_operating_conditions -model fast -temperature 85 -voltage 1200 update_timing_netlist  #change device speed grade and set operating conditions set_operating_conditions -speed 3 -model slow -temperature 0 -voltage 1100 update_timing_netlist</pre>				
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>		
	TCL_OK	0	INFO: Operation successful		

*continued...*

TCL_ERROR	1	ERROR: Cannot set operating conditions for timing netlist created from XML file. Run create_timing_netlist.
TCL_ERROR	1	ERROR: Both the -temperature and -voltage options and their values are required.
TCL_ERROR	1	ERROR: The ability to select multiple operating conditions at once has been disabled.
TCL_ERROR	1	ERROR: Values entered did not match any valid operating conditions. Available operating conditions are: <string>
TCL_ERROR	1	ERROR: The <string> device family does not support set_operating_conditions command.
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
TCL_ERROR	1	ERROR: Illegal value: <string>. Specify an integer ranging from -99999999 to 99999999 for the option -voltage
TCL_ERROR	1	ERROR: Unsupported option: <string>.

### 3.1.29.79. timing\_netlist\_exist (::quartus::sta)

The following table displays information for the timing\_netlist\_exist Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	timing_netlist_exist [-h   -help] [-long_help]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Checks if the timing netlist exists. Returns 1, if the timing netlist exists. Returns 0, otherwise.		
<b>Example Usage</b>	<pre>if {[!timing_netlist_exist]} {     create_timing_netlist }</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

### 3.1.29.80. update\_timing\_netlist (::quartus::sta)

The following table displays information for the update\_timing\_netlist Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	update_timing_netlist [-h   -help] [-long_help] [-dynamic_borrow] [-full] [-no_borrow] [-recompute_borrow]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-dynamic_borrow	Use time borrowing values that are correct for the current clock constraints	
	-full	Forces creation of an updated timing netlist to ensure correctness	

*continued...*

	-no_borrow	Turn off all time borrowing	
	-recompute_borrow	Recompute optimal time borrowing values	
<b>Description</b>	Updates and applies SDC commands to the timing netlist. The update_timing_netlist command expands and validates generated clocks, warns about sources in the design that require clock settings, identifies and removes combinational loops, and warns about undefined input/output delays. Most Tcl commands (e.g., report_timing) automatically update the timing netlist when necessary. You can use the update_timing_netlist command explicitly to control when updating occurs, or to force a full update using the -full option. The update_timing_netlist command can also be used to control time borrowing behavior. Time borrowing is a technique whereby certain flip-flops in certain device families are allowed to have signals that arrive late (thus improving upstream slack), at the expense of downstream slack. The amount of time borrowing allowed at each flip-flop is hardware-dependent. By default, optimal time borrowing values are computed at the end of the Fitter (Finalize) stage (if enabled by your compilation settings), and these are the values you will see in the timing reports. To turn off time borrowing support, use the -no_borrow option. This is not recommended, as it may result in significantly pessimistic timing results. If you have changed clock constraints after compiling your design, pre-computed optimal time borrowing values may no longer be valid. Time borrowing will be turned off for the changed clocks, resulting in pessimistic timing. To get optimal results once again, run update_timing_netlist with the -recompute_borrow option. This may take significant time on large designs, but the results will be saved and available the next time you run update_timing_netlist without any time borrowing options. The time borrowing optimization algorithm has a few limitation - for example, it never borrows any time on sources of cross-clock transfers, sources of paths with set_max_delay or set_max_skew constraints, or in any clock domain containing at least one level-sensitive latch. If your design is correctly constrained, you may overcome these limitations and get better timing results with the -dynamic_borrow option, which calculates time borrowing amounts based on your actual clock constraints (rather than optimizing for highest FMax within each clock domain). Note that -dynamic_borrowing is not recommended for overconstrained designs.		
<b>Example Usage</b>	<pre>project_open top create_timing_netlist read_sdc update_timing_netlist  report_timing -to_clock clk1 report_timing -to_clock clk2  delete_timing_netlist project_close</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.

### 3.1.29.81. use\_timing\_analyzer\_styleEscaping (::quartus::sta)

The following table displays information for the use\_timing\_analyzer\_styleEscaping Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393		
<b>Syntax</b>	use_timing_analyzer_styleEscaping [-h   -help] [-long_help] [-off] [-on]		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-off	Disable this setting.	
	-on	Enable this setting.	
<b>Description</b>	Use Timing Analyzer-style escaping. (Timing Analyzer-style escaping is enabled by default. The values used to create a collection, whether explicitly using a collection command or implicitly as a value specified as a "-from", "-to", or similar option to various SDC and report commands, are a Tcl list of wildcards. This includes a single name with an exact match. The value must follow standard Tcl		
	<i>continued...</i>		

	<p>substitution rules for Tcl lists and "string match" as described below, unless using Timing Analyzer-style escaping (default). For special characters such as '\$', the character must be escaped using a single '\' character to prevent Tcl from interpreting the word after '\$' as a Tcl variable, such as: Clk\\$Signal. A '\' character itself must be escaped with another '\' as in the '\$' case, must be escaped again for the Tcl list, and must be escaped yet again for Tcl "string match." The final result is eight '\' characters, such as: Clk\\\\\\\\\\\$Signal. Using Tcl "list" eliminates one level of escaping, since it will escape any '\' characters automatically for the Tcl list, such as: [list Clk\\\\\\\$Signal] Using '{' and '}' characters also eliminates the need for one or two levels of escaping, since '{' and '}' prevent string substitution in the contents, such as: [List {Clk\\\$Signal}] {{Clk\\\$Signal}} The use_timing_analyzer_styleEscaping option, which is on by default, allows the user to specify a name containing '\' characters with only two '\' characters in all cases, such as: Clk\\\$Signal. The extra '\' characters required for Tcl list string substitution and "string match" are added automatically by the Timing Analyzer. To disable Timing Analyzer style string escaping, call "use_timing_analyzer_styleEscaping -off" before adding any timing constraints or exceptions.</p>									
<b>Example Usage</b>	<pre>project_open top use_timing_analyzer_styleEscaping -on create_timing_netlist set res [get_cells my_test special_\\reg] query_collection \$res -all  delete_timing_netlist project_close</pre>									
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Code Name</b></th><th><b>Code</b></th><th><b>String Return</b></th></tr> </thead> <tbody> <tr> <td>TCL_OK</td><td>0</td><td>INFO: Operation successful</td></tr> <tr> <td>TCL_ERROR</td><td>1</td><td>ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.</td></tr> </tbody> </table>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>	TCL_OK	0	INFO: Operation successful	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
<b>Code Name</b>	<b>Code</b>	<b>String Return</b>								
TCL_OK	0	INFO: Operation successful								
TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.								

### 3.1.29.82. write\_sdc (::quartus::sta)

The following table displays information for the write\_sdc Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::sta on page 393	
<b>Syntax</b>	write_sdc [-h   -help] [-long_help] [-expand] [-history] [-valid_exceptions] <file_name>	
<b>Arguments</b>	-h   -help	Short help
	-long_help	Long help with examples and possible return values
	-expand	Generate SDC file by expanding the macros
	-history	Reports full history of assignments
	-valid_exceptions	Generate SDC file containing only valid timing exceptions for debugging purposes
	<file_name>	Name of output file
<b>Description</b>	Generates an SDC file with all current constraints and exceptions. When you use the -expand option, derive_clocks, derive_pll_clocks, derive_lvds_clocks and derive_clock_uncertainty macros are expanded to corresponding sdc assignments before they are written to a file. If you do not use the -expand option, these macros are preserved. Use the -valid_exceptions option to generate an SDC file that contains only valid timing exceptions. Run the report_exceptions command with the -valid option to see all the valid timing exceptions in your design.	
<b>Example Usage</b>	<pre>project_new test create_timing_netlist create_clock -period 10 -name clk10 clk set_multicycle_path -from [get_cells a] -to [get_cells b] update_timing_netlist  report_timing  write_sdc my_sdc_file.sdc</pre>	

*continued...*

	delete_timing_netlist project_close		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: Clock manager is not up-to-date. Run update_timing_netlist to generate the latest clock manager.
	TCL_ERROR	1	ERROR: Timing netlist does not exist. Use create_timing_netlist to create a timing netlist.
	TCL_ERROR	1	ERROR: Open failed: <string>

### 3.1.30. ::quartus::stp

The following table displays information for the **::quartus::stp** Tcl package:

<b>Tcl Package and Version</b>	::quartus::stp 1.0
<b>Description</b>	This package contains the set of Tcl functions for acquiring Signal Tap data from the Intel device.
<b>Availability</b>	This package is loaded by default in the following executables:  quartus_stp quartus_stp_tcl
<b>Tcl Commands</b>	<a href="#">close_session</a> (::quartus::stp) on page 491 <a href="#">export_data_log</a> (::quartus::stp) on page 492 <a href="#">open_session</a> (::quartus::stp) on page 493 <a href="#">run</a> (::quartus::stp) on page 493 <a href="#">run_multiple_end</a> (::quartus::stp) on page 495 <a href="#">run_multiple_start</a> (::quartus::stp) on page 495 <a href="#">stop</a> (::quartus::stp) on page 496

#### 3.1.30.1. close\_session (::quartus::stp)

The following table displays information for the **close\_session** Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::stp</a> on page 491		
<b>Syntax</b>	<code>close_session [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
<b>Description</b>	Saves the current session to the existing Signal Tap File (.stp).		
<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #capture data to log named log1, timeout after 5 seconds if no trigger occurs if { [catch {run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 - data_log log_1 -timeout 5} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" }  #close signaltap session close_session</pre>		
Return Value	Code Name	Code	String Return
	TCL_OK	0	INFO: Operation successful

*continued...*

TCL_OK	0	INFO: Session has been saved in Signal Tap File and closed
TCL_ERROR	1	ERROR: Can't open Signal Tap File for writing. Make sure Signal Tap File exists, has write permission, and is not currently being used by another program.
TCL_ERROR	1	ERROR: Session has not been opened. Make sure a session is open before attempting to close it.

### 3.1.30.2. export\_data\_log (::quartus::stp)

The following table displays information for the `export_data_log` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::stp</a> on page 491		
<b>Syntax</b>	<code>export_data_log [-h   -help] [-long_help] [-clock_period &lt;clock period&gt;] [-data_log &lt;data log&gt;] -filename &lt;export file name&gt; [-format &lt;export format&gt;] [-instance &lt;instance&gt;] [-signal_set &lt;signal set&gt;] [-trigger &lt;trigger&gt;]</code>		
<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-clock_period &lt;clock period&gt;</code>	The file name of the exported file	
	<code>-data_log &lt;data log&gt;</code>	Name of data log to be exported	
	<code>-filename &lt;export file name&gt;</code>	The file name of the exported file	
	<code>-format &lt;export format&gt;</code>	File format of the exported file	
	<code>-instance &lt;instance&gt;</code>	Name of instance that defines data log	
	<code>-signal_set &lt;signal set&gt;</code>	Name of signal set that defines data log	
	<code>-trigger &lt;trigger&gt;</code>	Name of trigger that defines data log	
<b>Description</b>	Exports the specified data log from the current open session into another file in different format. If a data log is not explicitly specified, the last active one is used. The supported file formats are Comma Separated Value file (.csv), Value Change Dump file (.vcd), and Table file (.tbl).		
<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #capture data to log named log1, timeout after 5 seconds if no trigger occurs if { [catch {run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 -timeout 5} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" }  #export data into a VCD file export_data_log -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 -filename log_1.vcd -format vcd  #close signaltap session close_session</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Data has been acquired successfully

*continued...*

TCL_ERROR	1	ERROR: Error occurred when the specified data log was exported to a file.
TCL_ERROR	1	ERROR: Instance, signal set, or trigger does not exist. Make sure the instance, signal set, and trigger exist in the Signal Tap File.
TCL_ERROR	1	ERROR: Session has not been opened. Make sure a session is open before attempting to close it.

### 3.1.30.3. open\_session (::quartus::stp)

The following table displays information for the `open_session` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::stp on page 491		
<b>Syntax</b>	<code>open_session [-h   -help] [-long_help] -name &lt;.stp file name&gt;</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
	-name <.stp file name>	Signal Tap File (.stp) name	
<b>Description</b>	Opens a session from the specified Signal Tap File (.stp).		
<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #capture data to log named log1, timeout after 5 seconds if no trigger occurs if { [catch {run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 - data_log log_1 -timeout 5} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" }  #close signaltap session close_session</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Session has been opened from Signal Tap File
	TCL_ERROR	1	ERROR: Can't open Signal Tap File for reading. Make sure the Signal Tap File exists and has read permission.
	TCL_ERROR	1	ERROR: Session already open. Close session before attempting to open it again.
	TCL_ERROR	1	ERROR: Signal Tap File contains syntax error. Make sure the Signal Tap File is formatted correctly before opening. Intel recommends that you do not manually edit Signal Tap Files, but use the Signal Tap dialog boxes in the Quartus Prime GUI.

### 3.1.30.4. run (::quartus::stp)

The following table displays information for the `run` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::stp on page 491		
<b>Syntax</b>	<code>run [-h   -help] [-long_help] [-bridge &lt;bridge&gt;] [-data_log &lt;data log&gt;] [-device_name &lt;device name&gt;] [-hardware_name &lt;hardware name&gt;] [-instance &lt;instance&gt;] [-signal_set &lt;signal set&gt;] [-timeout &lt;timeout&gt;] [-trigger &lt;trigger&gt;]</code>		

*continued...*

<b>Arguments</b>	<code>-h   -help</code>	Short help	
	<code>-long_help</code>	Long help with examples and possible return values	
	<code>-bridge &lt;bridge&gt;</code>	Bridge to use instead of the one specified in the stp file	
	<code>-data_log &lt;data log&gt;</code>	Name of data log to be recorded	
	<code>-device_name &lt;device name&gt;</code>	Device to use instead of the one specified in the stp file. Tcl command, <code>get_device_names</code> , can be used to obtain the valid hardware names	
	<code>-hardware_name &lt;hardware name&gt;</code>	JTAG programming hardware to use instead of the one specified in the stp file. Tcl command, <code>get_hardware_names</code> , can be used to obtain the valid hardware name	
	<code>-instance &lt;instance&gt;</code>	Name of instance that defines data acquisition	
	<code>-signal_set &lt;signal set&gt;</code>	Name of signal set that defines data acquisition	
	<code>-timeout &lt;timeout&gt;</code>	Timeout period for data acquisition in seconds	
	<code>-trigger &lt;trigger&gt;</code>	Name of trigger that defines data acquisition	
<b>Description</b>	Starts data acquisition with the specified conditions in the session and saves data into the specified data log within the timeout period.		
<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #capture data to log named log1, timeout after 5 seconds if no trigger occurs if { [catch {run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 - data_log log_1 -timeout 5} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" }  #close signaltap session close_session</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Data has been acquired successfully
	TCL_ERROR	1	ERROR: JTAG chain in use. Wait for JTAG communication to finish and run again.
	TCL_ERROR	1	ERROR: Data acquisition stopped unexpectedly. Make sure device is stable and run again.
	TCL_ERROR	1	ERROR: Trigger not compatible with device. Download a design with the current SRAM Object File after recompiling.
	TCL_ERROR	1	ERROR: Instance, signal set, or trigger does not exist. Make sure the instance, signal set, and trigger exist in the Signal Tap File.
	TCL_ERROR	1	ERROR: Session has not been opened. Make sure a session is open before attempting to close it.
	TCL_ERROR	1	ERROR: Trigger did not occur in timeout period. Make sure trigger conditions are valid and/or increase timeout period.

### 3.1.30.5. run\_multiple\_end (::quartus::stp)

The following table displays information for the `run_multiple_end` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::stp on page 491		
<b>Syntax</b>	<code>run_multiple_end [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> </ul>		Short help
	<ul style="list-style-type: none"> <li>-long_help</li> </ul>		Long help with examples and possible return values
<b>Description</b>	Defines the end of a set of "run" commands. This command is used when multiple instances of data acquisition are started simultaneously. Add "run_multiple_start" before the set of "run" commands that specify data acquisition. Add this command after the set of commands. If "run_multiple_end" is not included, the "run" commands do not execute.		
<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #start acquisition of instance auto_signaltap_0 and auto_signaltap_1 at the same time #calling run_multiple_end will start all instances run after run_multiple_start call run_multiple_start run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 - timeout 5 run -instance auto_signaltap_1 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 - timeout 5 if { [catch {run_multiple_end} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" }  #close signaltap session close_session</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Multiple instances of data acquisition ended successfully
	TCL_ERROR	1	ERROR: Data acquisition stopped unexpectedly. Make sure device is stable and run again.
	TCL_ERROR	1	ERROR: Run multiple instances has not been started. Use run_multiple_start before using run_multiple_end.
	TCL_ERROR	1	ERROR: Session has not been opened. Make sure a session is open before attempting to close it.

### 3.1.30.6. run\_multiple\_start (::quartus::stp)

The following table displays information for the `run_multiple_start` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::stp on page 491		
<b>Syntax</b>	<code>run_multiple_start [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<ul style="list-style-type: none"> <li>-h   -help</li> </ul>		Short help
	<ul style="list-style-type: none"> <li>-long_help</li> </ul>		Long help with examples and possible return values
<b>Description</b>	Defines the start of a set of "run" commands. This command is used when multiple instances of data acquisition are started simultaneously. Add this command before the set of "run" commands that specify data acquisition. Add "run_multiple_end" after the set of commands. If "run_multiple_end" is not included, the "run" commands do not execute.		

*continued...*

<b>Example Usage</b>	<pre>#opens signaltap session open_session -name stpl.stp  #start acquisition of instance auto_signtap_0 and auto_signtap_1 at the same time #calling run_multiple_end will start all instances run after run_multiple_start call run_multiple_start run -instance auto_signtap_0 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 - timeout 5 run -instance auto_signtap_1 -signal_set signal_set_1 -trigger trigger_1 -data_log log_1 - timeout 5 if { [catch {run_multiple_end} err_msg] {     # Timeout event is thrown as TCL exception     puts "ERROR: \$err_msg" } }  #close signaltap session close_session</pre>		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_OK	0	INFO: Multiple instances of data acquisition started
	TCL_ERROR	1	ERROR: Run multiple instances has not ended. Use run_multiple_end to complete an active call to run_multiple_start before using run_multiple_start again.
	TCL_ERROR	1	ERROR: Session has not been opened. Make sure a session is open before attempting to close it.

### 3.1.30.7. stop (::quartus::stp)

The following table displays information for the `stop` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::stp</a> on page 491						
<b>Syntax</b>	<code>stop [-h   -help] [-long_help]</code>						
<b>Arguments</b>	<table border="1"> <tr> <td><code>-h   -help</code></td> <td>Short help</td> </tr> <tr> <td><code>-long_help</code></td> <td>Long help with examples and possible return values</td> </tr> </table>			<code>-h   -help</code>	Short help	<code>-long_help</code>	Long help with examples and possible return values
<code>-h   -help</code>	Short help						
<code>-long_help</code>	Long help with examples and possible return values						
<b>Description</b>	Stops all data acquisition.						
<b>Example Usage</b>	<pre>stop</pre>						
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>				
	TCL_OK	0	INFO: Operation successful				
	TCL_OK	0	INFO: Data acquisition has stopped				
	TCL_ERROR	1	ERROR: No data acquisition was started. Data acquisition must be in progress before stopping.				

### 3.1.31. ::quartus::tdc

The following table displays information for the `::quartus::tdc` Tcl package:

<b>Tcl Package and Version</b>	::quartus::tdc 1.0		
<b>Description</b>	This package contains the set of Tcl functions for obtaining information from the Timing Analyzer.		
<b>Availability</b>	This package is loaded by default in the following executables: <code>qpro</code> <code>quartus</code>		
	<i>continued...</i>		

	quartus_fit quartus_map quartus_pow quartus_sta quartus_syn
<b>Tcl Commands</b>	<a href="#">is_place (::quartus::tdc) on page 497</a> <a href="#">is_plan (::quartus::tdc) on page 497</a> <a href="#">is_post_route (::quartus::tdc) on page 498</a>

### 3.1.31.1. is\_place (::quartus::tdc)

The following table displays information for the `is_place` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::tdc on page 496</a>		
<b>Syntax</b>	<code>is_place [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns true when called from Fitter during the placer. (Only supported in Quartus Prime Pro Edition)		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: This command is not supported in the current software edition.

### 3.1.31.2. is\_plan (::quartus::tdc)

The following table displays information for the `is_plan` Tcl command:

<b>Tcl Package and Version</b>	Belongs to <a href="#">::quartus::tdc on page 496</a>		
<b>Syntax</b>	<code>is_plan [-h   -help] [-long_help]</code>		
<b>Arguments</b>	-h   -help	Short help	
	-long_help	Long help with examples and possible return values	
<b>Description</b>	Returns true when called from Fitter during the plan. (Only supported in Quartus Prime Pro Edition)		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: This command is not supported in the current software edition.

### 3.1.31.3. `is_post_route` (::quartus::tdc)

The following table displays information for the `is_post_route` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::tdc on page 496		
<b>Syntax</b>	<code>is_post_route [-h   -help] [-long_help]</code>		
<b>Arguments</b>	<code>-h   -help</code> <code>-long_help</code>		Short help Long help with examples and possible return values
<b>Description</b>	Returns true when called from Fitter after router has completed or when post-fitting delays are annotated. (Only supported in Quartus Prime Pro Edition)		
<b>Example Usage</b>	This command currently contains no example usage.		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful
	TCL_ERROR	1	ERROR: No project is currently open. Open an existing project or create a new project.
	TCL_ERROR	1	ERROR: This command is not supported in the current software edition.

### 3.1.32. ::quartus::uno

The following table displays information for the ::quartus::uno Tcl package:

<b>Tcl Package and Version</b>	::quartus::uno 1.0
<b>Description</b>	This package contains no general description.
<b>Availability</b>	This package is loaded by default in the following executable: <code>quartus_synth</code>
<b>Tcl Commands</b>	<code>uno::write_verilog</code> (::quartus::uno) on page 498

#### 3.1.32.1. `uno::write_verilog` (::quartus::uno)

The following table displays information for the `uno::write_verilog` Tcl command:

<b>Tcl Package and Version</b>	Belongs to ::quartus::uno on page 498	
<b>Syntax</b>	<code>uno::write_verilog [-h   -help] [-long_help] -design &lt;design&gt; [-module &lt;module&gt;] [-output_file &lt;output_file&gt;] [-split_modules]</code>	
<b>Arguments</b>	<code>-h   -help</code>	Short help
	<code>-long_help</code>	Long help with examples and possible return values
	<code>-design &lt;design&gt;</code>	Design for writing verilog
	<code>-module &lt;module&gt;</code>	Module for writing verilog
	<code>-output_file &lt;output_file&gt;</code>	Output file for writing verilog
	<code>-split_modules</code>	Flag for writing each module to one file
<b>Description</b>	UNO write verilog	

*continued...*

<b>Example Usage</b>	uno::write_verilog -design \$design		
<b>Return Value</b>	<b>Code Name</b>	<b>Code</b>	<b>String Return</b>
	TCL_OK	0	INFO: Operation successful

## 3.2. Tcl Commands and Packages Revision History

The following revision history applies to this chapter:

**Table 13. Document Revision History**

Document Version	Intel Quartus Prime Version	Changes
2021.10.04	21.3	<ul style="list-style-type: none"><li>• Updated for latest Tcl commands and packages support.</li></ul>
2021.03.29	21.1	<ul style="list-style-type: none"><li>• Updated for latest Tcl commands and packages support.</li></ul>
2020.12.14	20.4	Initial release of <i>Tcl Commands and Packages</i> reference chapter.

## 4. Intel Quartus Prime Pro Edition User Guide Scripting Archives

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
21.1	<a href="#">Intel Quartus Prime Pro Edition User Guide Scripting</a>
20.4	<a href="#">Intel Quartus Prime Pro Edition User Guide Scripting</a>
19.1	<a href="#">Intel Quartus Prime Pro Edition User Guide Scripting</a>
18.1	<a href="#">Intel Quartus Prime Pro Edition User Guide Scripting</a>
18.0	<a href="#">Scripting User Guide Intel Quartus Prime Pro Edition</a>

## A. Intel Quartus Prime Pro Edition User Guides

---

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Pro Edition FPGA design flow.

### Related Information

- [Intel Quartus Prime Pro Edition User Guide: Getting Started](#)  
Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- [Intel Quartus Prime Pro Edition User Guide: Platform Designer](#)  
Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- [Intel Quartus Prime Pro Edition User Guide: Design Recommendations](#)  
Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.
- [Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)  
Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- [Intel Quartus Prime Pro Edition User Guide: Design Optimization](#)  
Describes Intel Quartus Prime Pro Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).
- [Intel Quartus Prime Pro Edition User Guide: Programmer](#)  
Describes operation of the Intel Quartus Prime Pro Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.
- [Intel Quartus Prime Pro Edition User Guide: Block-Based Design](#)  
Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.

- [Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration](#)  
Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Simulation](#)  
Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec\*, Cadence\*, Siemens EDA, and Synopsys that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis](#)  
Describes support for optional synthesis of your design in third-party synthesis tools by Siemens EDA, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools](#)  
Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin\*.
- [Intel Quartus Prime Pro Edition User Guide: Debug Tools](#)  
Describes a portfolio of Intel Quartus Prime Pro Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or "tapping") signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, system debugging toolkits, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- [Intel Quartus Prime Pro Edition User Guide: Timing Analyzer](#)  
Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- [Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization](#)  
Describes the Intel Quartus Prime Pro Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- [Intel Quartus Prime Pro Edition User Guide: Design Constraints](#)  
Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- [Intel Quartus Prime Pro Edition User Guide: PCB Design Tools](#)  
Describes support for optional third-party PCB design tools by Siemens EDA and Cadence\*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- [Intel Quartus Prime Pro Edition User Guide: Scripting](#)  
Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.