Sentaurus™ Workbench User Guide

Version T-2022.03, March 2022

SYNOPSYS®

Contents

About This Guide

The Synopsys® Sentaurus™ Workbench tool is the primary graphical front end for the integration of TCAD Sentaurus simulation software into one environment.

Sentaurus Workbench provides a convenient framework to design, organize, and automatically run complete TCAD simulation projects. Its user interface drives various Synopsys simulation and visualization tools as well as third-party tools, and automates the execution of fully parameterized projects. Sentaurus Workbench also supports design-of-experiments, extraction and analysis of results, optimization, and uncertainty analysis. It has an integrated job scheduler to speed up simulations and takes full advantage of distributed, heterogeneous, and corporate computing resources.

For additional information, see:

- Documentation installed with the software package and available from the Sentaurus Workbench **Help** menu
- The TCAD Sentaurus release notes, available on the Synopsys SolvNetPlus support site (see Accessing SolvNetPlus on page 16)
- Documentation available on the SolvNetPlus support site

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
`Courier font`	Identifies text that is displayed on the screen or that you must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
Italicized text	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.
Key+Key	Indicates keyboard actions, for example, Ctrl+I (press the I key while pressing the Ctrl key).

Convention	Description
Menu > **Command**	Indicates a menu command, for example, **File** > **New** (from the **File** menu, choose **New**).

Customer Support

Customer support is available through the Synopsys SolvNetPlus support site and by contacting the Synopsys support center.

Accessing SolvNetPlus

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

To access the SolvNetPlus site:

1. Go to https://solvnetplus.synopsys.com.
2. Enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

Contacting Synopsys Support

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

- Go to the Synopsys Global Support Centers site on www.synopsys.com. There you can find email addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNetPlus site or the Synopsys Global Support Centers site and open a case (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an email message to:

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

1

Introduction to Sentaurus Workbench

This chapter provides an overview of Sentaurus Workbench.

Sentaurus Workbench Functionality

Sentaurus Workbench is the framework environment designed to simplify the use of Synopsys TCAD tools. It frees you from typing system commands for handling data files or starting applications. One of its main advantages is the possibility of parameterizing input files to run simulation groups automatically. The main features of Sentaurus Workbench include:

- The user interface simplifies the editing and handling of complex simulation projects, and the flexible open tool interface makes it possible to plug in third-party tools.
- It allows for the flexible configuration and storing of the view of the project simulation flow.
- Simulations can be organized into projects and folders, which provide a clear overview of the overall simulation environment.
- The project database is mapped to the underlying native file systems and allows robust file management in a multiuser distributed environment.
- You can set up tool flows with multiple instances of the same tool.
- Simulation parameters can be used in any input file, and the resulting simulation experiments can be edited before running the simulations.
- It is easy to build new simulation projects by copying parts of existing projects.
- Many example projects are available to be copied and modified as required.
- You can perform design-of-experiments, optimization, and statistical analysis.

- The scheduler integrated into the user interface can schedule and monitor the running of simulation projects.
- There is easy access to backend scheduling systems to run large simulations in parallel on a network of workstations and computing clusters.

Sentaurus Workbench Projects

A project consists of a family of scenarios. Each scenario consists of several experiments where certain input variables take different values. Parameters can be introduced at any point in the simulation flow, from the process to the device simulation phases.

A project exists as a directory in the file system. If a directory contains a `.project` file, this indicates to Sentaurus Workbench that the current directory is a *project directory*.

A parameterized project is represented as a tree structure, shown in the Family Tree part of the project view, which is derived from a simulation flow and a combination of all the parameter values. Each level shown in the Family Tree corresponds to a simulation phase, as defined in the simulation flow.

In the Family Tree, each node has a unique number – the node key (`<nkey>`). A real node represents the end of a simulation phase and holds the output of the corresponding tool instance. Sentaurus Workbench associates output to a node by adding the prefix `n<nkey>_` to all the output file names of the tool used.

You need to distinguish between real and virtual simulation phases. Real simulation phases correspond to the execution of tool instances, and virtual simulation phases are introduced by parameters and do not lead to any tool execution. In a real simulation phase, there are as many tool instances as nodes in the corresponding tree level. Each tool instance is characterized by a combination of parameter values that define the path from the root node to the tool instance node.

The main attribute of a project is its runtime editing mode, which defines the editing and running policy for that project. You can choose either the Locked mode or the Editable mode (the default mode). Projects in the Locked mode have the maximum level of automation and consistency; however, you are limited with regard to applying changes to a running project. In contrast, projects in the Editable mode have maximum flexibility, while Sentaurus Workbench partially delegates to you the responsibility of maintaining consistency between input data and simulation results. You can change the runtime editing mode at any time in the preferences (see Changing the Default Runtime Editing Mode of Projects on page 33).

Compatibility With Previous Versions

Sentaurus Workbench can load, edit, and run projects created in previous versions of the tool. Sentaurus Workbench makes all of the necessary conversions automatically.

Note:

There is no guarantee that you can edit new projects using earlier versions of Sentaurus Workbench.

Starting Sentaurus Workbench

You must set up environment variables before starting Sentaurus Workbench.

Setting Up Environment Variables

You must set up the following environment variables before starting Sentaurus Workbench:

`STDB`	The directory where all user projects reside.
`STRELEASE`	The version of Sentaurus Workbench. If not specified, the default is the *current* version.
`STROOT`	The location where the Synopsys TCAD software is installed.

The following example illustrates the settings in a `.cshrc` file:

```
setenv STROOT /home/user/ST
setenv STRELEASE T-2022.03
setenv STDB $HOME/DBtest
```

Launching Sentaurus Workbench From the Command Line

You can launch Sentaurus Workbench from the command line by specifying:

```
swb [<options>] [<directory> | <file>]
```

where:

- `<options>` can be any of the following:
  - `-a`: Initializes the Sentaurus Workbench Advanced mode that includes the **Optimization** and **Calibration** menus.
  - `-b`: Initializes the Sentaurus Workbench Basic mode that does not include the **Optimization** and **Calibration** menus.
  - `-default`: Resets preferences to the default settings.
  - `-h`: Displays help information.

- -nowait: Switches off license queuing.

  Without this option, if you launch Sentaurus Workbench when no license is available, Sentaurus Workbench queues for the license from the license manager and waits until a license is available. If you launch Sentaurus Workbench with the -nowait option, it will not queue for a license. If no license is available, Sentaurus Workbench exits.

- -plugin null: Switches off standard plug-ins.
- -v: Displays the version of Sentaurus Workbench.
- -verbose: Displays all messages.

- <directory>: Specify a Sentaurus Workbench project directory.
- <file>: Specify a compressed file with a Sentaurus Workbench project.

Note:

If you launch Sentaurus Workbench without either the -a or -b option, it opens in the Basic mode.

Examples

Launch Sentaurus Workbench Basic mode:

```
swb &
```

```
swb -b &
```

Launch Sentaurus Workbench Basic mode and load a project directory:

```
swb /u/jbrown/sample_app/myproject &
```

TCAD Sentaurus Tutorial: Simulation Projects

The TCAD Sentaurus Tutorial provides various projects demonstrating the capabilities of Sentaurus Workbench.

To access the TCAD Sentaurus Tutorial:

1. Open Sentaurus Workbench by entering the following on the command line: swb
2. From the menu bar of Sentaurus Workbench, choose **Help** > **Training** or click T on the toolbar.

Alternatively, to access the TCAD Sentaurus Tutorial:

1. Go to the `$STROOT/tcad/current/Sentaurus_Training` directory.

   The `STROOT` environment variable indicates where the Synopsys TCAD distribution has been installed.

2. Open the `index.html` file in your browser.

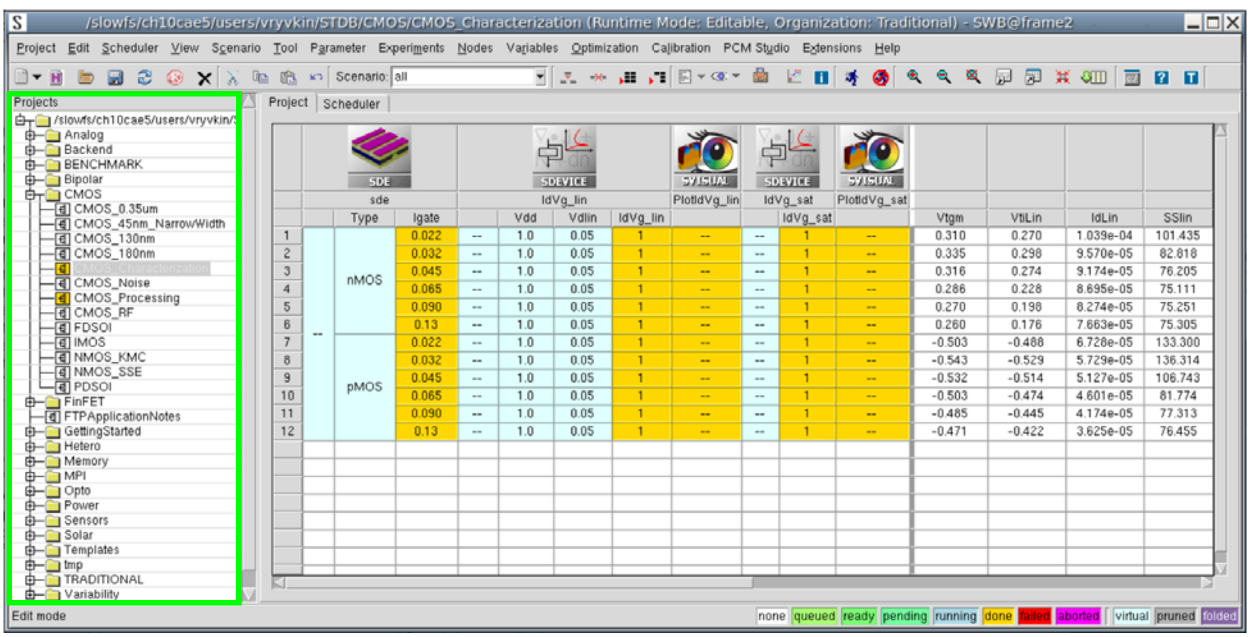
User Interface

The main window of Sentaurus Workbench consists of the projects browser and tabs for the Project Editor (**Project** tab) and the Scheduler (**Scheduler** tab).

Projects Browser

You can manage projects in the projects browser (the Projects panel) (see Figure 1). The projects are organized hierarchically as a tree. The tree displays the current file system as specified by the setting of the `STDB` environment variable.

Figure 1 Main window with projects browser (green box) and Project Editor view showing traditional horizontal flow orientation



Directories are identified as Sentaurus Workbench project directories by the presence of a `.project` file, and Sentaurus Workbench displays project directories like files with the or **H** icon for projects with traditional and hierarchical organization, respectively. The color of the icon reflects the status of the project (see Figure 17 on page 66).

The projects browser shows a global view of the project database of the user as a hierarchy of folders and projects. It features a tree representation to navigate through this hierarchy, to open and close folders, and to load projects, and for diverse operations on entire projects and folders, such as copying and moving projects. Additional hierarchies of folders and projects can be attached to the projects browser.

Note:

Any folders under a project directory are not displayed. Sentaurus Workbench does not work if there is a `.project` file anywhere above `STDB`.

Viewing Directories

The directories can be expanded or collapsed, allowing you to view the subdirectories and project folders below directories. You can navigate the directories using keyboard and mouse operations.

Right-click the Projects panel to display the context menu.

Attaching Root Directories

While you can specify the file system directory where all your projects reside using the `STDB` environment variable, you can also include other directories at the same level, in the projects browser.

Such a directory is a *root directory*, which typically contains a collection of folders and projects, and it can also be a project directory.

To add a root directory to the projects browser:

1. In the Projects panel, right-click and choose **Folder** > **Attach Root**.
2. In the Open Directory dialog box, navigate to the directory to be added.
3. Click **OK**.

Note:

Any directories above a root directory must not contain project directories or `.project` files, which Sentaurus Workbench interprets as project directories.

Updating the Status of Directories

In the projects browser, the status of the directory structure, attached root directories, and projects is updated regularly. The interval for updates is set in the preferences. However, you can update a project directory or root directory manually at any time.

To update the status of a directory:

► Right-click the directory and choose **Refresh**, or press the F5 key.

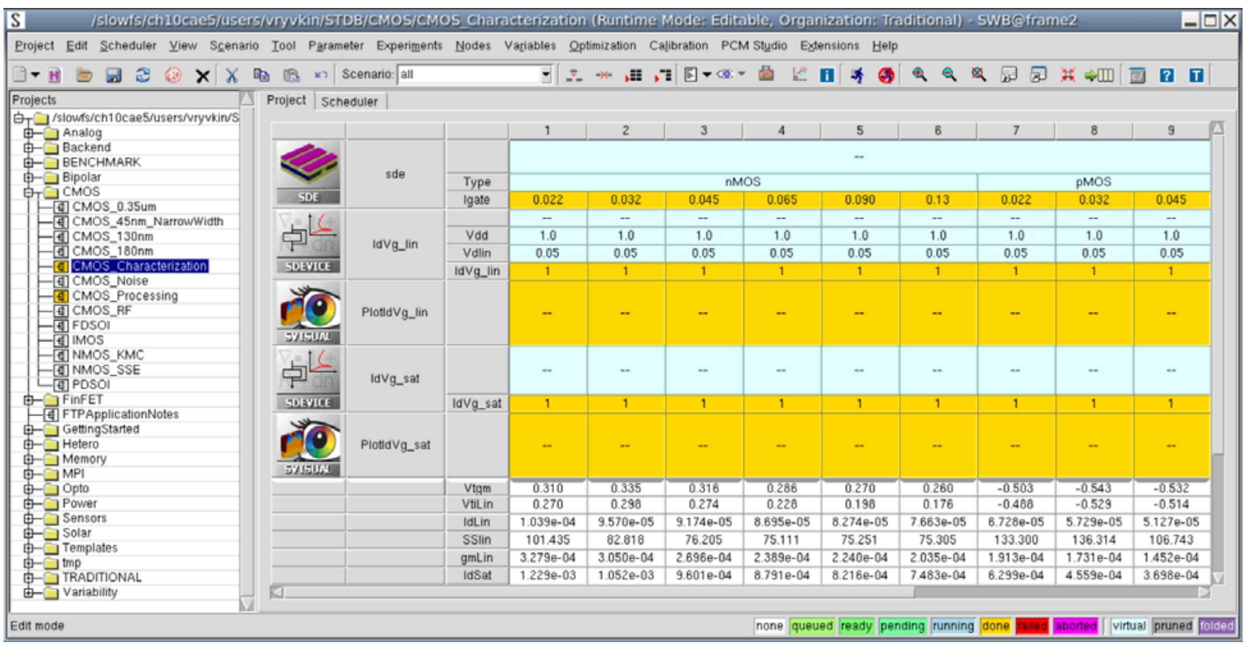
Project Editor

The Project Editor allows you to access, organize, and edit a database of simulation projects on the **Project** tab. The simulation flow can be oriented horizontally (see Figure 1 on page 22) or vertically (see Figure 2).

The **Project** tab displays an individual project as a table of experiments and simulation results. The rows from left to right in the horizontal flow orientation (see Figure 1) or columns from top to bottom in the vertical flow orientation (see Figure 2) represent the simulation flow followed by extracted results.

The simulation flow is the sequence of tools running the simulations steps, split by parameters. Columns in the horizontal orientation (or rows in the vertical orientation) represent different experiments and their corresponding parameter and variable values. You can add, remove, and modify experiments, and control the running of experiments.

Figure 2 Main window with Project Editor view showing vertical flow orientation



Scheduler

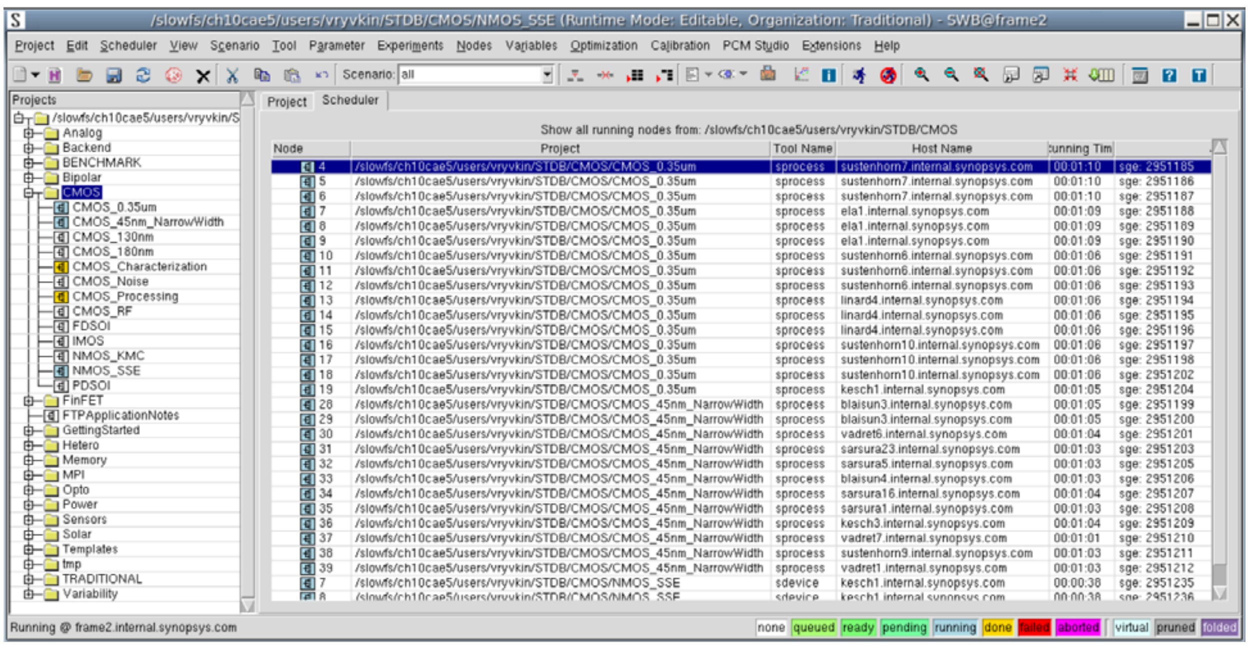
You can use the Scheduler to submit, terminate, and monitor the simulation jobs of a project. You can also define scheduling queues and job-mapping restrictions. The Scheduler gives you an overview of the running jobs and their distribution on the local area network.

To open the Scheduler:

► Choose **Scheduler** > **Show Scheduler**, or click the **Scheduler** tab (see Figure 3).

The **Scheduler** tab displays a table of running jobs with scheduling information, such as running time and running host. Only jobs belonging to the selected project or folder are shown. By default, Sentaurus Workbench shows the running jobs of the currently open project.

Figure 3 Main window of Sentaurus Workbench showing the Scheduler tab



Utilities

This section describes the different utilities available in Sentaurus Workbench.

gcleanup

This utility cleans up a project and performs all of the cleanup operations including the renumbering of the tree (see Cleaning Up Projects From the Command Line on page 217).

genopt

The Optimization Framework is a modern batch tool available in the Sentaurus Workbench Advanced mode. It is designed to extract efficiently general information about TCAD simulations. It provides a set of tools for optimization, screening, and sensitivity analysis. For more information, see the *Sentaurus™ Workbench Optimization Framework User Guide*.

gjob

This utility manages the execution of individual jobs and controls the evaluation of the job prologue and epilogue, and the running of the corresponding simulation tool (see Launching a Specific Job on page 182).

gpythonsh

This utility is a general-purpose Python shell you can use at the command prompt and as a Python tool in your simulation flow. It is binary compatible with the TCAD Python platform.

All Python modules coming with the TCAD Python are accessible in gpythonsh. One of these modules is swbpy2, which provides the Python API of Sentaurus Workbench, such as tree manipulation. Other modules deliver the simulation capabilities of TCAD simulators, which can be imported into gpythonsh. In addition, gpythonsh is the binary for TCAD to SPICE tools (Mystic, Garand VE, and RandomSpice).

In addition, you can run gpythonsh in the IPython mode. IPython is a powerful interactive Python shell, which provides a rich toolkit to help you optimize using Python interactively. The IPython module is included in TCAD Python. To launch gpythonsh with the IPython module, use the following command:

```
gpythonsh -m IPython
```

gsub

This utility consists of a simple command to submit jobs to the Scheduler for execution. It also constitutes the interface of Sentaurus Workbench to different internal or external batch systems (see Submitting Jobs to Queues on page 181).

gtclsh

This utility is a tool command language (Tcl) shell that has been extended with all of the internal commands of Sentaurus Workbench, such as tree manipulation.

spp

This utility is the Sentaurus Workbench preprocessor that prepares a project for execution, which includes:

- Calculating an optimum execution graph from the simulation tree to use parallel computing while enforcing start–completion job interdependencies.
- Generating actual tool input files from user-provided templates to differentiate experiments.

See Preprocessing Projects on page 206.

swbdiag

This utility is used to troubleshoot issues originating in customer environments that might affect the behavior of Sentaurus Workbench (see Appendix E on page 334).

swblm

This daemon process is a dispatcher of interprocess communications between the Sentaurus Workbench components `swb`, `gjob`, `gsub`, and `spp`. Sentaurus Workbench starts this process automatically when needed. You do not need to terminate a running daemon process or to start a new one manually (see Appendix E on page 334).

2

Managing Projects

This chapter describes the operations that can be performed on projects and directories using the projects browser.

Traditional and Hierarchical Project Organizations

Sentaurus Workbench supports projects with traditional and hierarchical organizations. For details, see Chapter 12 on page 284.

Sentaurus Workbench automatically recognizes the project organization and handles the project appropriately.

Every Sentaurus Workbench project stores its organization (traditional or hierarchical) in the project file `.organization`. The project organization is stored when you create a new project. If this file is omitted, then Sentaurus Workbench assumes that the project has traditional organization.

Sentaurus Workbench allows you to convert a project with traditional organization to hierarchical organization (see Converting Project Organization on page 31).

Sentaurus Workbench does not allow you to convert a project with hierarchical organization to traditional organization. However, you can do it manually.

Creating New Projects

You must instruct Sentaurus Workbench about the organization of a new project.

To create a new project with traditional organization:

► Choose **Project** > **New** > **Traditional Project**, or press Ctrl+N, or click the **Create New Project** toolbar button.

To create a new project with hierarchical organization:

► Choose **Project** > **New** > **Hierarchical Project**, or click the **Create New Project** toolbar button.

Sentaurus Workbench creates a new project directory with a temporary name. You will be requested to specify the permanent project name when you save the project.

Note:

Sentaurus Workbench prohibits the use of the following characters in project names, including full paths: / \ ~ * ? $! " < > : [ ] { } = | ; <tab> <space>

In addition, it is not recommended to use any of the following characters in project names: , @ # ( ) ' ` + & ^ %

Creating Folders

Note:

You cannot create folders inside project directories.

To create a folder:

1. Select a directory that is not a project directory.
2. Choose **Project** > **New** > **Folder**, or right-click and choose **Folder** > **New Folder**.
3. Enter the name of the new folder.
4. Press the Enter key.

Note:

Sentaurus Workbench prohibits the use of the following characters in folder names, including full paths: / \ ~ * ? $! " < > : [ ] { } = | ; <tab> <space>

In addition, it is not recommended to use any of the following characters in folder names: , @ # ( ) ' ` + & ^ %

Opening Projects

To open a project:

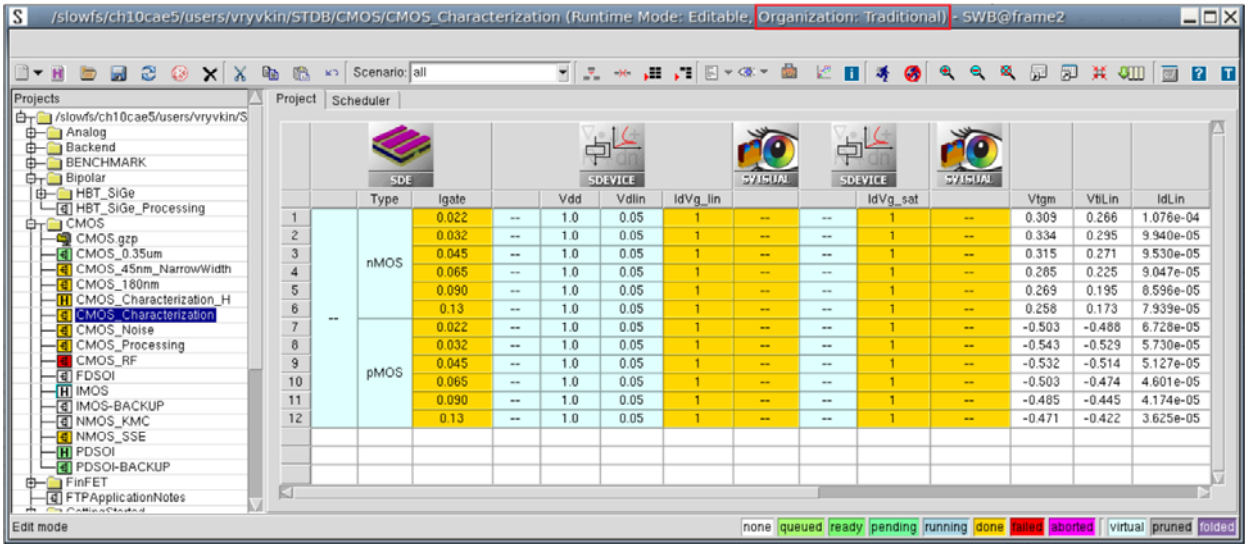
► Double-click the project in the projects browser.

Alternatively, to open a project:

1. Select the project in the projects browser.
2. Press the Enter key, or double-click the selection, or right-click and choose **Open**.

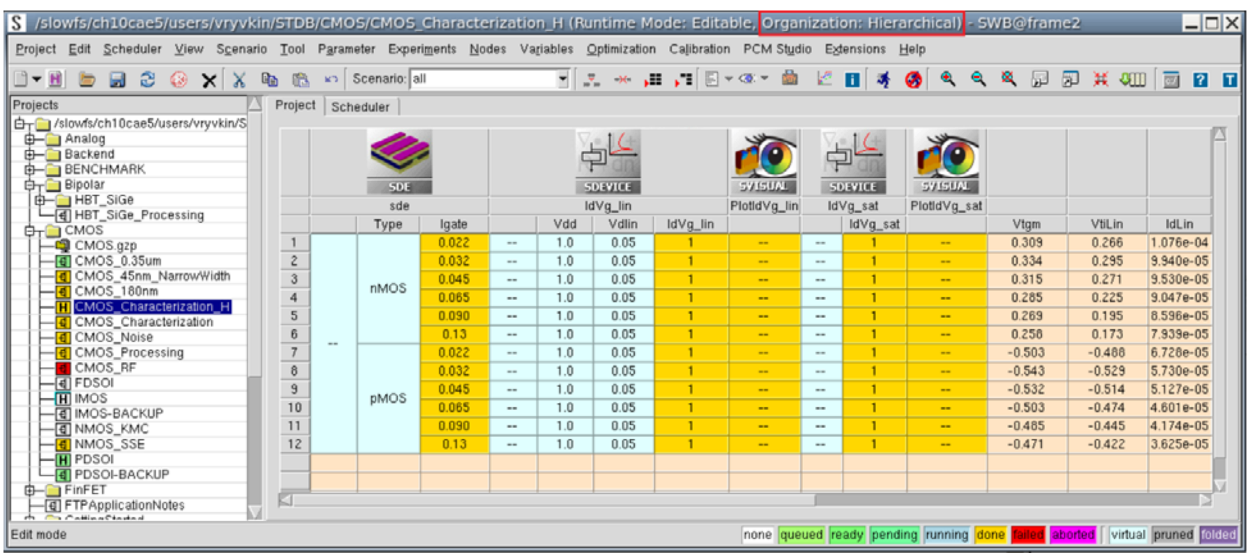
When you open a project, Sentaurus Workbench automatically recognizes the project organization and handles the project appropriately. Additional text in the title bar of the main window indicates the organization of a project (see Figure 4).

Figure 4 Title bar of Sentaurus Workbench main window indicating a project with traditional organization (red box)



If a project has hierarchical organization, then the background color of Sentaurus Workbench differs from that in traditional mode (see Figure 5). By default, it is bisque.

Figure 5 Title bar of Sentaurus Workbench main window indicating a project with hierarchical organization (red box)



To change the background and foreground colors of the main window of Sentaurus Workbench in hierarchical mode:

1. Choose **Edit** > **Preferences**.

   The SWB Preferences dialog box opens.

2. Expand **Project** > **Organization** > **GUI Settings for Hierarchical Projects**.
3. Change the background and foregrounds colors as required.
4. Click **Apply**.

Converting Project Organization

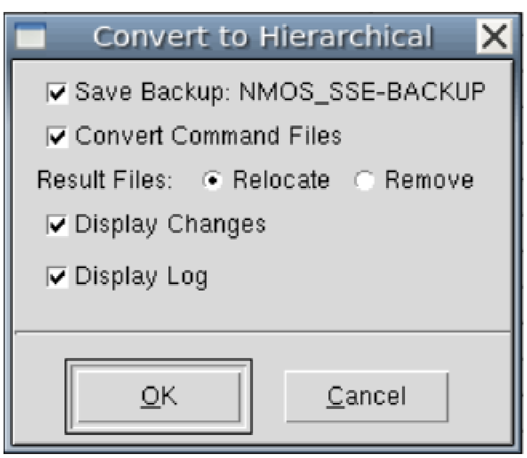
Note:

Sentaurus Workbench does not provide a converter for forward compatibility. When you convert a project to hierarchical organization, you cannot reconvert it automatically to traditional organization. However, with manually applied changes (removing the `.organization` file or changing tool input command files, for example), you still can work with the project in traditional mode.

To convert the organization of a project from traditional to hierarchical:

1. Select the project in the Projects panel.
2. Choose **Project** > **Operations** > **Convert to Hierarchical**, or click the **Convert Project**  button.

   The Convert to Hierarchical dialog box opens.



3. Select the required options.

   The default options are configured according to the settings in user preferences, under **Project** > **Organization** > **Converter From Traditional Project Organization**:

   ◦ Select **Save Backup** to save a backup copy of the original project with the given name in the same directory where the original project resides.

   ◦ Select **Convert Command Files** to parse tool input command files and to apply changes required for hierarchical project organization.

   Note:
   When converting from traditional to hierarchical organization, Sentaurus Workbench might not convert all your setups. Manual changes to tool input command files might be required.

   For **Result Files**, you select the required action for existing log files and node files if the original project has been preprocessed or executed:

   ◦ Select **Relocate** for Sentaurus Workbench to recognize existing files and to move them into the corresponding results hierarchy.

   ◦ Select **Remove** for Sentaurus Workbench to delete these files and to clean up the project.

   If you select **Display Changes**, then Sentaurus Workbench launches a *diff* tool for every tool input command file changed during conversion.

   If you select **Display Log**, then Sentaurus Workbench displays the conversion log file (`results/logs/convert.log`) with a report of the project conversion. This file might contain warnings with references to the instructions Sentaurus Workbench could not convert, in which case, you might need to make changes manually.

4. Click **OK**.

Changing the Runtime Editing Mode of a Project

Note:
Projects must be in the writable area to change the runtime editing mode. Furthermore, you cannot change the runtime editing mode of a running project.

To change the runtime editing mode of a project:

1. Select a project in the Projects panel.
2. Right-click the project, choose **Project** > **Runtime Editing Mode**, and choose either:

   ◦ **Locked** for the Locked mode

  - **Editable** for the Editable mode

See Locked Runtime Editing Mode on page 185 and Editable Runtime Editing Mode on page 186.

Changing the Default Runtime Editing Mode of Projects

The Editable mode is the default mode for projects. However, you can configure Sentaurus Workbench to set up the runtime editing mode automatically for new projects and any old projects that do not have a runtime editing mode selected.

To change the default runtime editing mode of projects:

1. Choose **Edit** > **Preferences**, or press the F12 key.
2. In the SWB Preferences dialog box, expand **Project** > **Runtime Editing Mode** > **Default Mode**.
3. Select either **Locked** or **Editable**.
4. Click **Apply**.

Copying Projects or Folders

You can copy multiple projects or folders.

Note:

Projects and folders cannot be copied inside other projects.

If a project is copied over its original project, the new project is created in the same folder with `Copy_of_` prefixed to the project name. If a folder is copied over its original folder, the new folder with all its contents is created in the same parent folder with `Copy_of_` prefixed to the folder name.

To copy projects or folders:

1. In the Projects panel, select the required projects.
2. Press Ctrl+C, or right-click and choose **Copy**.
3. Navigate to the required destination folder or keep the selection.
4. Press Ctrl+V, or right-click and choose **Paste**.

Searching for Files and Projects

You can search for files and Sentaurus Workbench projects in the `STDB` root directory. You can configure Sentaurus Workbench to find files and projects based on either a keyword or a file pattern. In addition, Sentaurus Workbench can identify files by content.

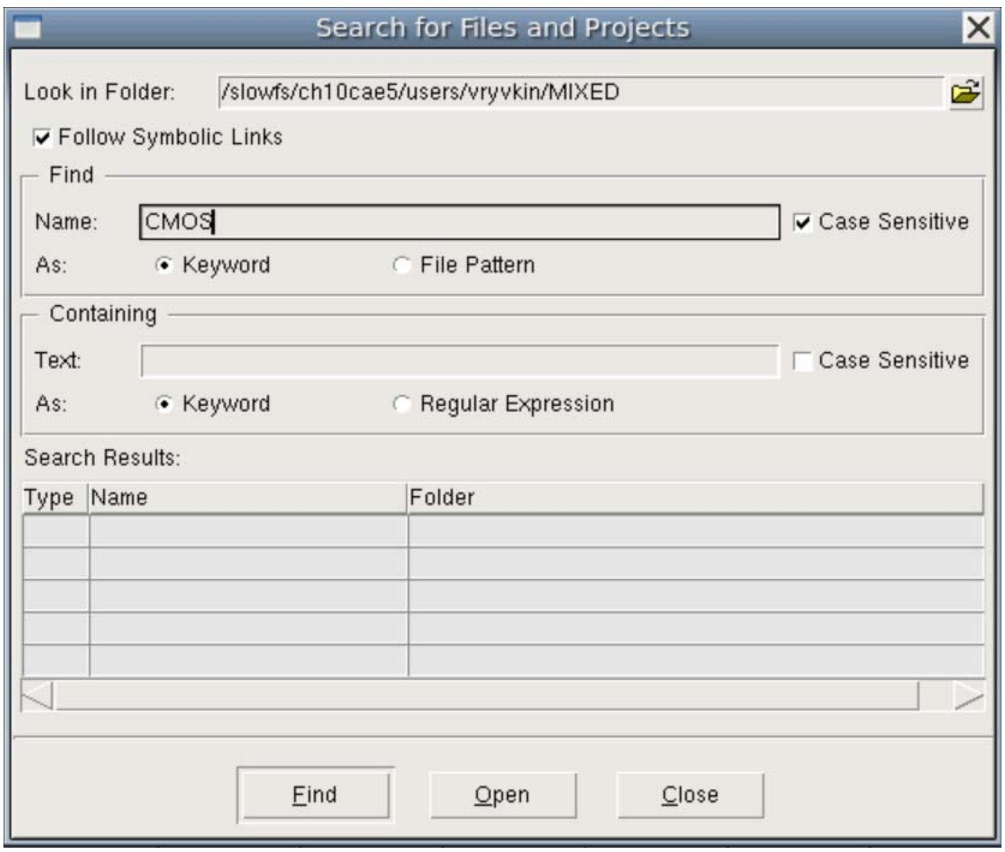
To search for files and projects:

1. In the Projects panel, select the folder you want to search.

   By default, Sentaurus Workbench searches for files in the `STDB` root directory.

2. Choose **Project** > **Search**, or right-click the folder and choose **Folder** > **Search**.

   The Search for Files and Projects dialog box opens. The **Follow Symbolic Links** option is selected by default.



3. In the **Name** field, enter a string to search for.

   You can specify multiple file patterns as a space-delimited list.

4. Select **Case Sensitive** if required.

   Select this option if Sentaurus Workbench must differentiate between uppercase and lowercase letters when looking for files that match the specified string.

5. Select one of the following options:

   - **Keyword**: Sentaurus Workbench looks for files that have the specified string in their names.

   - **File Pattern**: Sentaurus Workbench looks for files that match the file pattern in the **Name** field. You must use standard UNIX file pattern syntax (see Using File Patterns on page 36).

   Note:
   The keyword `abc` is equivalent to the file pattern `*abc*`.

6. In the **Text** field, enter a string that a file must contain as content, if required.

7. Select **Case Sensitive** if required.

   Select this option if Sentaurus Workbench must differentiate between uppercase and lowercase letters when checking file content according to the string in the **Text** field.

8. Select one of the following options:

   - **Keyword**: A matching file must contain at least one instance of the string in the **Text** field.

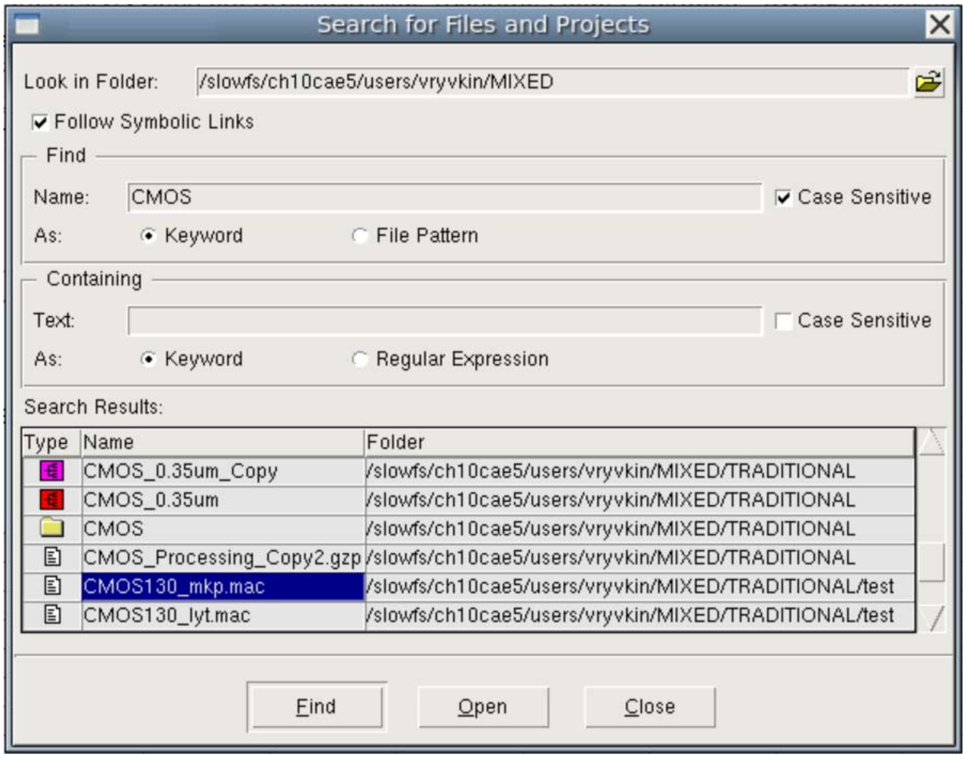
   - **Regular Expression**: Sentaurus Workbench checks files for the string in the **Text** field in the form of a regular expression. You must use standard UNIX regular expression syntax for the shell command `grep` (see Using Regular Expressions on page 37).

   Note:
   Sentaurus Workbench checks the content of text files only. It does not inspect binary files, such as TDR files.

9. Click **Find**.

   The button label changes to **Stop**. In the case of a long-running search, you can stop the process by clicking **Stop**. After the search is completed or if you have stopped it explicitly, the button label reverts to **Find**.

10. Select the required file, folder, or Sentaurus Workbench project in the **Search Results** table.



11. Click **Open**.

    The behavior of Sentaurus Workbench, when you click **Open**, depends on your selection in the **Search Results** table:

    ◦ A file in the Type or Name column: Sentaurus Workbench launches either Sentaurus Visual or the text editor to view the file content.

    ◦ A folder in the Type or Name column: Sentaurus Workbench opens a command prompt in the selected folder.

    ◦ A project in the Type or Name column: Sentaurus Workbench opens the project.

    ◦ A parent folder in the Folder column: Sentaurus Workbench opens a command prompt in the selected folder.

12. Click **Close** to close the dialog box.

Using File Patterns

You must use standard UNIX syntax when using file patterns to search for files. You use these file patterns when launching UNIX commands such as `find`, `locate`, and `ls`.

A file pattern is a string that can contain special characters, which are known as wildcards or metacharacters (see Table 1).

Table 1 Special characters for specifying file patterns

Character	Description
`*`	Matches any sequence of zero or more characters.
`?`	Matches any single character.
`[chars]`	Matches exactly one character, which is a member of `chars`. This is called a *character class*. As shorthand, `chars` can contain ranges, which consist of two characters with a dash between them. For example: • The range `[a-b]` specifies a sequence of any characters between `a` and `b` (inclusive). • The class `[a-z0-9_]` matches a lowercase letter, a number, or an underscore. You can negate a class by placing an exclamation mark `!` or a caret `^` immediately after the opening bracket. For example: • The class `[^A-Z@]` matches any character except an uppercase letter or an at symbol.
`\x`	Matches the character `x`, and removes the special meaning of the character `x` that follows it. This works even in character classes.

Sentaurus Workbench does not support subpatterns, which are allowed in the UNIX file pattern syntax. The following UNIX example matches any sequence of zero or more characters:

```
{a,b,...}
```

Instead, you can specify multiple file patterns in the **Name** field. For example, the UNIX file pattern:

```
?abc*.{log,txt,out}
```

is equivalent to the following list of patterns:

```
?abc*.log  ?abc*.txt  ?abc*.txt
```

Using Regular Expressions

You must use standard UNIX regular expression syntax for the shell command `grep`. Table 2 presents examples of the `grep` command with regular expressions. You can specify a regular expression directly in the **Text** field of the Search for Files and Projects dialog box.

Table 2 Regular expressions using grep command

Example	Description
`grep smug files`	Search *files* for lines with `smug`.
`grep '^smug' files`	Search for `smug` at the start of a line.
`grep 'smug$' files`	Search for `smug` at the end of a line.
`grep '^smug$' files`	Search for lines containing only `smug`.
`grep '^\^s' files`	Search for lines starting with `^s` (the backslash is used to escape the caret).
`grep '[Ss]mug' files`	Search for `Smug` or `smug`.
`grep 'B[oO][bB]' files`	Search for `Bob`, `BoB`, `BOb`, or `BOB`.
`grep '^$' files`	Search for blank lines.
`grep '[0-9][0-9]' files`	Search for pairs of numeric digits.

For details of the `grep` syntax, refer to the GNU Grep 3.0 regular expressions section at https://www.gnu.org/software/grep/manual/html_node/Regular-Expressions.html.

Linking Projects

You might want to link two different projects for various reasons:

- To establish execution dependency between projects
- To refer to the simulation results of one project in another project without executing the referred nodes
- To do both

In this section, such projects are called *parent and child projects* for demonstration purposes only.

You can link projects with different project organizations (see Chapter 12 on page 284).

The Bridge tool is used to link projects and must be added to the flow of a child project. Depending on the reason for linking projects, the command file of the Bridge tool might look differently.

Note:

Sentaurus Workbench supports only one Bridge tool in a project. In other words, you can link a child project to only one parent project.

Syntax of the Command File of the Bridge Tool

The command file of the Bridge tool specifies the name of a parent project, the execution dependency between the parent and child projects, and the nodes from the parent project to be linked to a child project.

The Bridge tool supports the following instructions specified in its command file `bridge_tcl.cmd`:

```
## Bridge node does not need to be executed
#noexec

## Absolute or relative path to the parent project
set Parent <path-to-parent-project>

## Execution dependency - should the child project start
## after the parent project is executed successfully?
set waitParent <yes|no>  ; # default: no

## Linking experiments - should the child project link experiments
## of the parent project?
set linkExperiments <yes|no>  ; # default: yes

## To which tools from the parent project should the child project
## link (takes effect only if linkExperiments=yes)?
set ptools <list-of-tools>  ; # default: empty list

## To which variables from the parent project should the child
## project link (takes effect only if linkExperiments=yes)?
set pvars <list-of-variables>  ; # default: empty list

## To which scenario from the parent project should the child project
## link (takes effect only if linkExperiments=yes)?
set PStoSync <name-of-scenario>  ; # default: all

## Verbosity level
set infoLevel <number>  ; # 0 - minimum (default), 3 - maximum
```

Note:

There is no need to execute the nodes of the Bridge tool, since the linking occurs automatically during project preprocessing and execution. This is the reason why the `#noexec` preprocessor directive is used in the Bridge command file.

Linking Projects for Execution Dependency

In this example, you instruct Sentaurus Workbench to establish an execution dependency to the given parent project without linking the simulation results.

The `Parent` setting contains the path to a parent project. The path can be either an absolute path or a relative path to a child project.

```
#noexec
set Parent "../../MyRepo/Deck22/SimpleMOS"
set waitParent yes
set linkExperiments no
```

The `waitParent` parameter defines a project execution dependency between a child project and a parent project. Launching a child project will start a parent project if the status of the parent project is not *done*. Only the remaining nodes in the parent project will be executed. After successful execution of the parent project, Sentaurus Workbench will launch the child project.

The `linkExperiments` parameter specifies not to reference the simulation results of the parent project in the child project.

Linking Projects to Reference Simulation Results

The parent project contains Sentaurus Workbench parameters, variables, and output files that you want to access from a child project.

```
#noexec
set Parent "../../MyRepo/Deck22/SimpleMOS"
set ptools "sprocess sdevice"
set pvars "tox"
set PStoSync all
set infoLevel 3
set waitParent yes      ; # only if you want an execution dependency
                        ; # to parent project
```

The next step is synchronizing the two projects to link them (see Synchronizing Dependent Projects on page 41). The Sentaurus Workbench parameters of a parent project are copied into the nodes of the Bridge tool after a child project is synchronized with a parent project.

The `ptools` setting is a list of tools in a parent project. Parameters defined for these tools in a parent project are copied into the Bridge parameters in a child project when the child and parent projects are synchronized. If you omit this setting, then Sentaurus Workbench copies the parameters of all tools of a parent project.

The `pvars` setting contains a list of variables of a parent project to copy to a child project. If you omit this setting, no variables are copied.

The `PStoSync` setting specifies the reference scenario in a parent project. It instructs Sentaurus Workbench to copy parameters with the values of the given scenario. If you omit this setting, all parameter values of a parent project will be copied to a child project, which is equivalent to `set PStoSync all`.

By default, this type of project linking does not set project execution dependency. However, you can set up the dependency explicitly by setting `waitParent` to `yes`. This setting does not affect linking experiments.

Sentaurus Workbench logs the project synchronization into the `syncwparent.log` file. The setting `infoLevel` sets the verbosity of output in the log file, which can range from 0 (no output) to 3 (the most detailed output). The default is 0.

The presence of the Bridge tool in a project flow instructs the preprocessor to support the following instructions for linking purposes:

- `@ppwd@`: Path to a parent project as defined in the Bridge tool input
- `@pprjorg@`: Project organization of a parent project: `traditional` or `hierarchical` (see Chapter 12 on page 284)
- `@ppwdout@`: Path to the directory where output files of a parent project are stored if project files are stored in a different location
- `@pnodesdir@`: Path to the directory where node folders of a parent project are stored
- `@plogsdir@`: Path to the directory where log files of a parent project are stored
- `@pnode|toolname@`: Matching node in a parent project that corresponds to the tool `toolname`

Synchronizing Dependent Projects

Different synchronization modes of parent and child projects are available under the **Tool** menu when the Bridge tool is selected:

- **Tool** > **Clean and Synchronize With Parent Project**

  This command performs the following operations:

  - Removes all parameters under the Bridge tool of a child project
  - Copies parameters of the required tools in a parent project to the Bridge tool of a child project (required tools are set using `ptools`)
  - Makes Sentaurus Workbench variables listed in `pvars` available in a child project after preprocessing or execution

Figure 6 on page 43 shows parent and child projects synchronized with the **Clean and Synchronize With Parent Project** command and executed.

- **Tool** > **Synchronize With Parent Project**

  This command performs the following operations:

  - Finds parameters of the required tools in a parent project that have not been synchronized yet and adds them to the Bridge tool of a child project (required tools are set using `ptools`)
  - Adds values of synchronized parameters from a parent project that are not yet in a child project
  - Makes Sentaurus Workbench variables listed in `pvars` available in a child project after preprocessing or execution

  This command is useful when new parameters or variations are added to a parent project while a child project already contains simulation results corresponding to the previous parent state. In that case, you might not want to clean up the child project.

  For an example, see Figure 7 on page 43.

- **Tool** > **Open Parent Project**

  This command closes the child project and opens the parent project in the same Sentaurus Workbench session.

- **Tool** > **Open Parent Project in New SWB Instance**

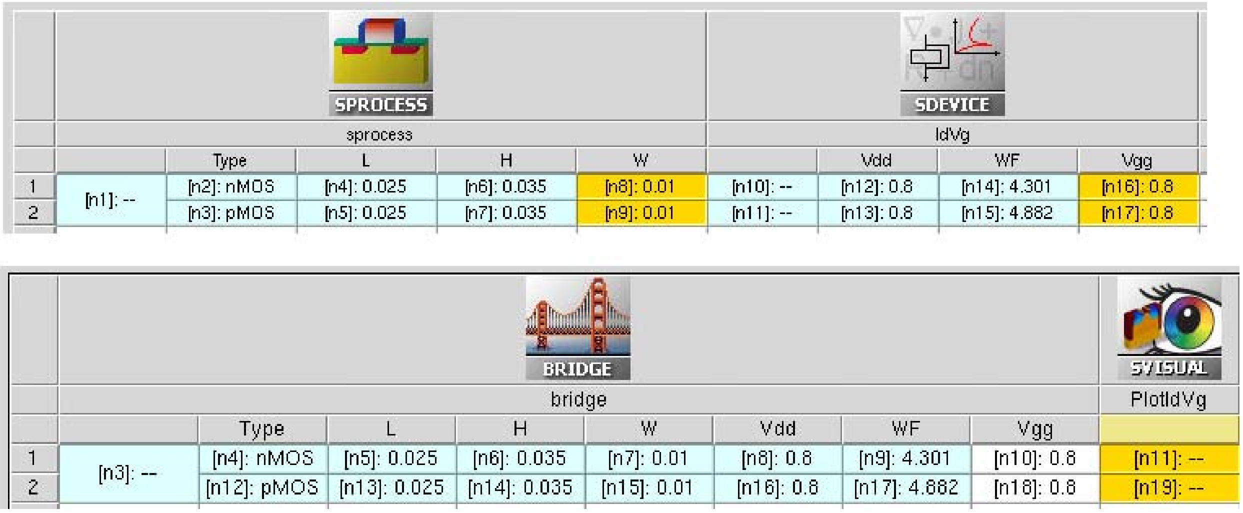
  This command launches a new Sentaurus Workbench session with the parent project opened. An additional Sentaurus Workbench license is required.

Note:

Sentaurus Workbench applies consistency checks to the `pvars` list against the parent project in both synchronization options. Sentaurus Workbench removes variables that do not exist in the parent project and adds variables that exist in the parent project. The change is saved in the Bridge command file.

If you decide to link an already synchronized child project to another parent project, then you should choose **Tool** > **Clean and Synchronize With Parent Project**. This command ensures that the child project is synchronized properly with the new parent project and does not contain traces of a previously linked parent project. The command **Tool** > **Synchronize With Parent Project** is designed for linking updated experiments in the same parent project without losing the results of already executed child projects.

Figure 6 (Top) Parent project and (bottom) child project after synchronization and execution of the child project

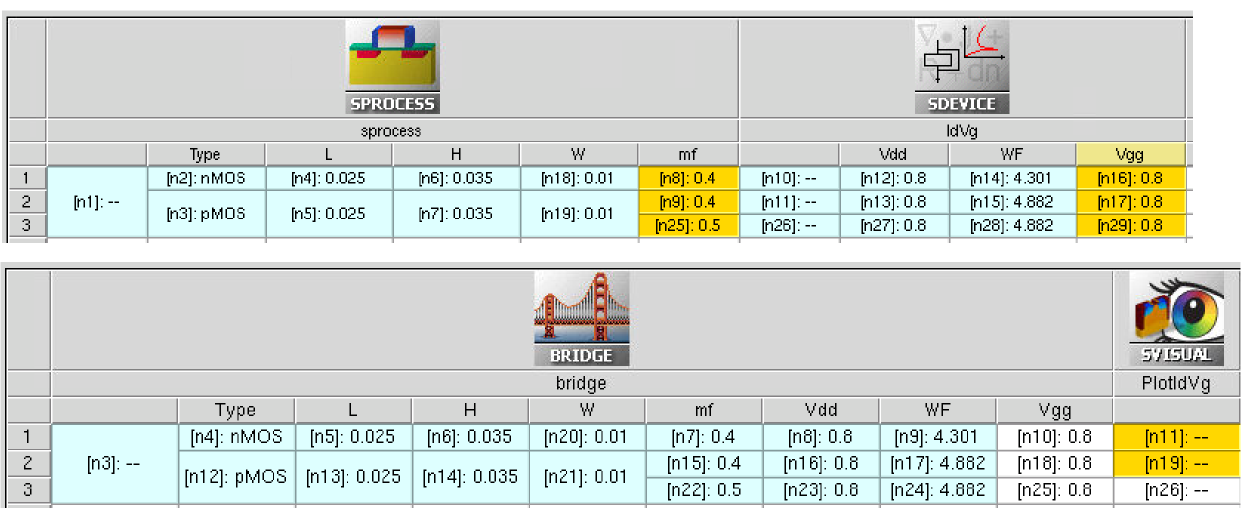


	SPROCESS sprocess					SDEVICE IdVg			
		Type	L	H	W		Vdd	WF	Vgg
1	[n1]: --	[n2]: nMOS	[n4]: 0.025	[n6]: 0.035	[n8]: 0.01	[n10]: --	[n12]: 0.8	[n14]: 4.301	[n16]: 0.8
2		[n3]: pMOS	[n5]: 0.025	[n7]: 0.035	[n9]: 0.01	[n11]: --	[n13]: 0.8	[n15]: 4.882	[n17]: 0.8

	BRIDGE bridge								SVISUAL PlotIdVg
		Type	L	H	W	Vdd	WF	Vgg	
1	[n3]: --	[n4]: nMOS	[n5]: 0.025	[n6]: 0.035	[n7]: 0.01	[n8]: 0.8	[n9]: 4.301	[n10]: 0.8	[n11]: --
2		[n12]: pMOS	[n13]: 0.025	[n14]: 0.035	[n15]: 0.01	[n16]: 0.8	[n17]: 4.882	[n18]: 0.8	[n19]: --

Figure 7 shows an example of using the **Synchronize With Parent Project** command. The parameter `mf` with values 0.4 (default) and 0.5 is added to the Sentaurus Process tool in the parent project. When the previously executed child project is synchronized without cleaning, the default `mf` parameter value (0.4) is added to the previously executed experiments. A new experiment corresponding to the new `mf` value (0.5) is added to the parent project.

Figure 7 (Top) Parent project and (bottom) child project synchronized with the parent project without cleaning the child project



	SPROCESS sprocess						SDEVICE IdVg			
		Type	L	H	W	mf		Vdd	WF	Vgg
1	[n1]: --	[n2]: nMOS	[n4]: 0.025	[n6]: 0.035	[n18]: 0.01	[n8]: 0.4	[n10]: --	[n12]: 0.8	[n14]: 4.301	[n16]: 0.8
2		[n3]: pMOS	[n5]: 0.025	[n7]: 0.035	[n19]: 0.01	[n9]: 0.4	[n11]: --	[n13]: 0.8	[n15]: 4.882	[n17]: 0.8
3						[n25]: 0.5	[n26]: --	[n27]: 0.8	[n28]: 4.882	[n29]: 0.8

	BRIDGE bridge									SVISUAL PlotIdVg
		Type	L	H	W	mf	Vdd	WF	Vgg	
1	[n3]: --	[n4]: nMOS	[n5]: 0.025	[n6]: 0.035	[n20]: 0.01	[n7]: 0.4	[n8]: 0.8	[n9]: 4.301	[n10]: 0.8	[n11]: --
2		[n12]: pMOS	[n13]: 0.025	[n14]: 0.035	[n21]: 0.01	[n15]: 0.4	[n16]: 0.8	[n17]: 4.882	[n18]: 0.8	[n19]: --
3						[n22]: 0.5	[n23]: 0.8	[n24]: 4.882	[n25]: 0.8	[n26]: --

Saving Projects

To save the current project in the same directory:

► Choose **Project** > **Save**, or press Ctrl+S.

To save a project with a different name or in a new directory (the entire contents of the project directory are saved to the new directory):

► Choose **Project** > **Save As** > **Project**.

To omit copying all of the preprocessed and node output files:

► Choose **Project** > **Save As** > **Clean Project**.

To create a new project based on selected experiments:

► Choose **Project** > **Save Selected Experiments As** > **Project**.

To create a new project based on selected experiments, but without copying the simulation results:

► Choose **Project** > **Save Selected Experiments As** > **Clean Project**.

Automatically Saving Projects

You can configure Sentaurus Workbench to save your projects automatically. Sentaurus Workbench periodically saves the currently opened project for your convenience.

Note:

Automatic saving takes effect only when projects are not running.

To switch on automatic saving of a project:

1. Choose **Edit** > **Preferences**, or press the F12 key.
2. In the SWB Preferences dialog box, expand **Project** > **Auto Save**.
3. Set **Project Auto Save Interval (min)** to a nonzero value specifying the interval in minutes.

   The default value of 0 means that this feature is switched off.
4. Click **Apply**.

Moving Projects or Folders

You can move projects and folders across directories by either a drag-and-drop operation or a cut-and-paste operation.

Note:

The following restrictions apply when moving projects or folders:

- You cannot move projects opened in the projects browser or running projects.
- You cannot move folders with any open or running project.
- You cannot place projects or folders inside projects.
- Projects without write permission are copied.

To move a project or folder using a drag-and-drop operation:

1. Select the project or folder in the Projects panel.
2. Drag the project or folder to the destination project or folder.

To move a project or folder using a cut-and-paste operation:

1. Select the project or folder in the Projects panel.
2. Press Ctrl+X, or right-click and choose **Cut**.
3. Navigate to the destination project or folder.
4. Press Ctrl+V, or right-click and choose **Paste**.

Renaming Projects or Folders

Note:

Operations on the Family Tree directly act on the file system. Therefore, any delete, move, or rename operation cannot be undone.

To rename a project or folder:

1. Select the project or folder.
2. Right-click and choose **Rename**.
3. Type the new name.
4. Press the Enter key.

Note:

A file name must contain *only* characters permitted by the operating system. Although the projects browser is configured to identify all invalid characters, extreme file names are likely to cause unpredictable behavior and might result in the loss of work.

Deleting Projects or Folders

You can delete multiple projects or folders.

Note:

This operation irreversibly removes project directories.

To delete a project or folder:

1. Select the project or folder in the Projects panel.
2. Choose **Edit** > **Delete**, or right-click and choose **Delete**.

Accessing Project Documentation

You can attach a documentation file to a project. The file format must be PDF, and the file name should be `greadme.pdf`. The file can contain any information about the project.

To access a documentation file in the default PDF viewer:

1. Select a project in the projects browser.
2. Choose **Project** > **Properties** > **Documentation**.

Exporting and Importing Projects

You can export projects and directories into a package.

Note:

You can redefine the default project export options in the SWB Preferences dialog box by expanding **Project** > **Export**.

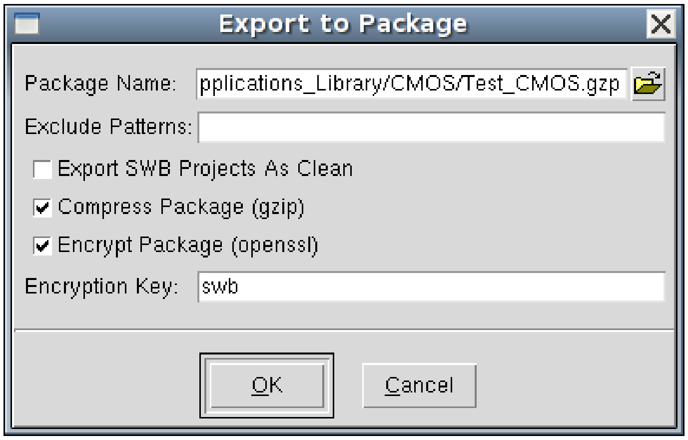
To export projects and directories:

1. Select one or more projects or data directories in the projects browser.

   To make multiple selections, hold the Ctrl key when clicking.

2. Choose **Project** > **Export**.

   The Export to Package dialog box opens.

3. In the **Package Name** field, enter the name and location of the package file.

4. In the **Exclude Patterns** field, enter the patterns of files and directories to be excluded from the package.

   Note:
   The exclude patterns are applied automatically (see Specifying Project Exclude File Patterns on page 219).

5. Select from the following options:

   ◦ **Export SWB Projects As Clean**: Packages all Sentaurus Workbench projects in the selected hierarchy as clean projects. The original projects are not cleaned up during packaging.

   ◦ **Compress Package (gzip)**: Compresses the package.

   ◦ **Encrypt Package (openssl)**: Encrypts the package.

6. If you selected **Encrypt Package (openssl)**, you must enter a key in the **Encryption Key** box (see Encrypted Packages on page 48).

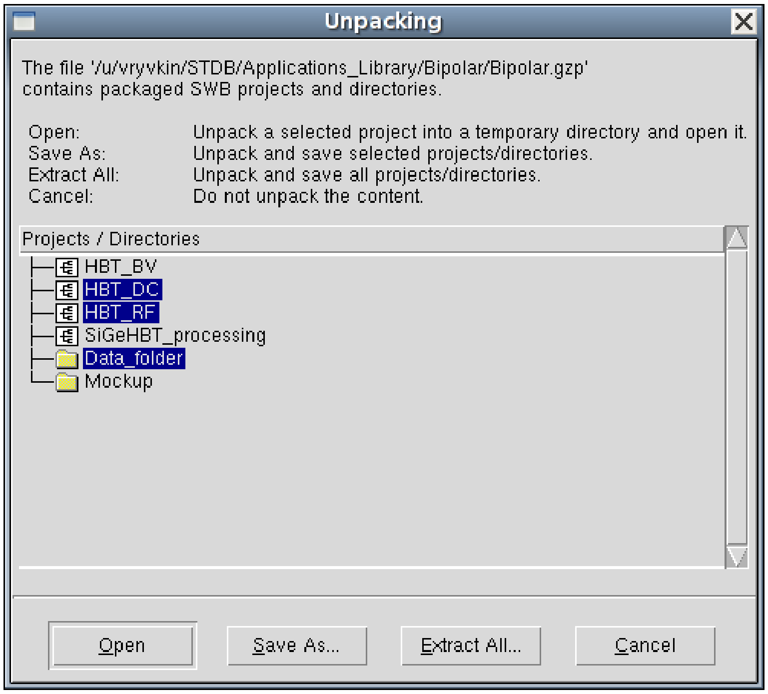
   Note:
   The following characters are prohibited in an encryption key:

   - Single quotation marks (')
   - Backslash (\)
   - Space (space, tabulator, newline)

7. Click **OK**.

To import projects and directories:

1. Double-click a `.gzp` or `.tar` file in the projects browser.
2. In the Unpacking dialog box, select one or more projects or directories.



3. Click **Save As** to unpack and store the selected items at the specified location.

   Click **Extract All** to unpack and store all the items from the file.

Encrypted Packages

Many users have difficulties using email to send packaged Sentaurus Workbench projects. Firewalls and email filters unpack the package, detect files with potentially dangerous extensions (`.cmd`), and block those files or even the entire email message.

Encryption allows your package to go through email filters without problems. When unpacking an encrypted package, Sentaurus Workbench uses a default encryption key. You can control the encryption key and package your projects with specific keys for confidential deliveries.

If the package has been encrypted with a nondefault key, Sentaurus Workbench asks you to enter this key when importing a packaged project.

Note:

With the default Synopsys encryption key `swb`, Sentaurus Workbench guarantees successful importing of packaged projects by another user. When you need additional security, package your project with a specific encryption key and communicate it to the recipient of the package.

You cannot import a packaged project that is encrypted with an unknown (or forgotten) key.

It is not recommended to send extremely large packages (exceeding 1 GB). There might be unexpected issues with unpacking such packages in different environments. In such cases, you should split the data into several packages that individually are less than 1 GB.

Configuring Behavior When Sentaurus Workbench Is Inactive

You might forget to exit Sentaurus Workbench, for example, before leaving for vacation or the weekend. As the result, inactive tools keep checked out licenses. You can configure Sentaurus Workbench to release licenses or to exit automatically if no user activity is registered for a given time.

If you configure Sentaurus Workbench to release licenses, then Sentaurus Workbench does not exit after the time-out. Instead, it releases licenses and displays a warning dialog box.

Click **Reclaim** to proceed working with Sentaurus Workbench, which keeps exactly the same state as when you left it. Click **Exit** to close Sentaurus Workbench.

Note:

You cannot work with Sentaurus Workbench until you click either **Reclaim** or **Exit**.

Figure 8 Inactive Tool dialog box

Configuring Sentaurus Workbench to Exit Automatically

To configure Sentaurus Workbench to exit automatically when there is no user activity:

1. Choose **Edit** > **Preferences**, or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Behavior on Inactivity** > **Inactivity Time-Out for SWB (hours)**.

3. Specify the number of hours of inactivity after which time Sentaurus Workbench will exit.

   If the time-out is set to 0, this feature is not activated.

4. For the **Alert Before Time-Out for SWB (min)** field, specify when a warning dialog box opens before the tool exits.

   If set to 0, no warning dialog box is shown before the tool exits.

   Note:
   Click the mouse button or press any key in the user interface of Sentaurus Workbench to reset the inactivity time-out.

5. In the **SWB Action After Time-Out** field, select **Exit**.

6. Click **Apply**.

As soon as the inactivity time-out expires, Sentaurus Workbench exits and prints a warning message to the terminal where it has been launched. For example:

```
WARNING: SWB has exited after 24 hours of user inactivity.
```

Note:
Unsaved changes in a project opened in Sentaurus Workbench are not stored when the tool exits.

Similarly, you can configure Sentaurus Visual to exit automatically when there is no user activity:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Behavior on Inactivity** > **Inactivity Time-Out for Visualization Tools (hours)**.

3. Specify the number of hours of inactivity after which time Sentaurus Visual will exit.

   If the time-out is set to 0, this feature is not activated.

4. For the **Alert Before Time-Out for Visualization (min)** field, specify when a warning dialog box opens before the tool exits.

If set to 0, no warning dialog box is shown before the tool exits.

Note:
Click the mouse button or press any key in the user interface of Sentaurus Visual to reset the inactivity time-out.

5. In the **Visualization Tool Action After Time-Out** field, select **Exit**.
6. Click **Apply**.

This exit setting applies to all Sentaurus Visual processes launched from and outside of Sentaurus Workbench.

Configuring Sentaurus Workbench to Release Licenses

To configure Sentaurus Workbench to release licenses when there is no user activity:

1. Choose **Edit** > **Preferences**, or press the F12 key.

   The SWB Preferences dialog box opens.
2. Expand **Behavior on Inactivity** > **Inactivity Time-Out for SWB (hours)**.
3. Specify the number of hours of inactivity after which time Sentaurus Workbench will release licenses.

   If the time-out is set to 0, this feature is not activated.
4. In the **SWB Action After Time-Out** field, select **Release License**.
5. Click **Apply**.

Similarly, you can configure Sentaurus Visual to release licenses automatically when there is no user activity:

1. Choose **Edit** > **Preferences**, or press the F12 key.

   The SWB Preferences dialog box opens.
2. Expand **Behavior on Inactivity** > **Inactivity Time-Out for Visualization Tools (hours)**.
3. Specify the number of hours of inactivity after which time Sentaurus Visual will exit.

   If the time-out is set to 0, this feature is not activated.
4. In the **Visualization Tool Action After Time-Out** field, select **Release License**.
5. Click **Apply**.

This setting for releasing licenses applies to all Sentaurus Visual processes launched from and outside of Sentaurus Workbench.

3

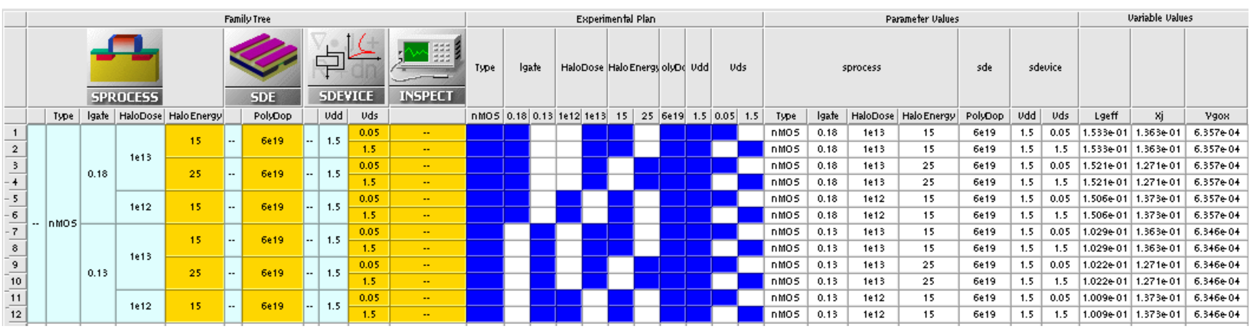
View Settings

This chapter describes how to modify and save the view settings for projects in Sentaurus Workbench.

View Settings for Projects

The **Project** tab consists of the following main sections: Family Tree, Experimental Plan, Parameter Values, and Variable Values (see Figure 9).

Figure 9 Different parts of the Project tab shown in the horizontal orientation



These sections, as well as other project view settings, can be configured by changing the default view options in the preferences or selecting the appropriate commands from the **View** menu. Your project view configuration is stored together with the project data. The next time you load the project, its view settings are used automatically.

You can configure the project view settings on both the user level and the project level (see Table 3).

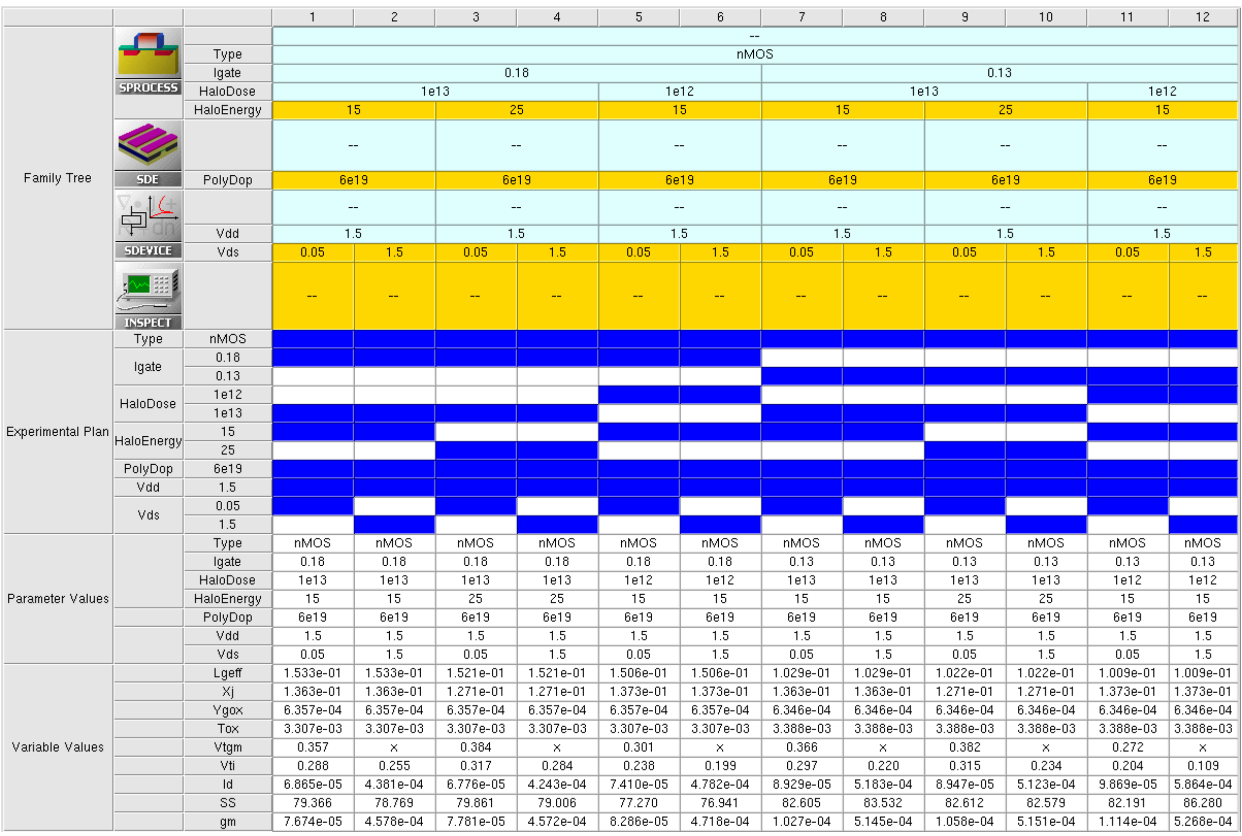
Project-level view settings override user-level settings. So, if a project does not have the `.database` file, the default view settings from the preferences are applied. Otherwise, the view settings from the `.database` file are used.

The project view on the **Project** tab can be oriented either horizontally or vertically. In this chapter, view settings for horizontally oriented projects are considered. For vertically oriented projects, the view settings apply in the same way.

Table 3 Setting project views at project and user levels

Level	Where to set	Description
Project	Commands are available from the **View** menu.	These view settings are stored in the `.database` file under the project directory and apply only to a specific project. The view settings are saved each time you save a project or switch to another project in the projects browser.
User	Choose **Edit** > **Preferences** to open the SWB Preferences dialog box. Expand **Table** > **Default View Options**.	These view settings apply to all projects of a given user, unless the project does not have project-level view settings in the `.database` file.

Figure 10 Different parts of the Project tab shown in the vertical orientation



Configuring the Default View Settings in Preferences

The default view settings are configured in the preferences (see Preferences on page 221). These settings apply to any project that does not have project-level view settings, that is, the project does not have the `.database` file in its directory.

Note:

It is recommended that you start by configuring the view settings in the preferences. Specify them so that they match the view requirements for most of your projects. You can customize these settings later for specific projects.

Restoring the Default View Settings in Preferences

The view settings stored in the `.database` file under the project directory override the settings in the preferences.

To restore the default view settings in the preferences for the currently opened project:

► Choose **View** > **Restore Default View Options**, or press Ctrl+8.

Configuring the Project Orientation

By default, Sentaurus Workbench displays projects in the horizontal orientation, where the tool flow is shown in the topmost row from left to right, and parameterized experiments are rows of the Family Tree (see Figure 9 on page 52).

Alternatively, you can display projects with a vertical orientation, where the tool flow is shown in the leftmost column from top to bottom, and parameterized experiments are columns of the Family Tree (see Figure 10 on page 53).

The project orientation depends on your preferences and the project structure. Some projects are more convenient to display horizontally, while other projects are best viewed in a vertical orientation.

To switch between horizontal and vertical orientations, either:

- Choose **View** > **Flow Orientation** > **Vertical**.
- Choose **View** > **Flow Orientation** > **Horizontal**.

The project orientation is stored in the `.database` file and is restored the next time you load the project.

Setting the Project View Mode

Sentaurus Workbench allows you to display projects in two modes:

- In *full mode* (default), the entire simulation flow is displayed (see Figure 1 on page 22).
- In *compact mode*, only varying parameterization parts of the simulation flow with extracted variables are displayed. All other parts of the flow are hidden. This mode allows you to focus on the active parameterization part of the project and can be useful for large design-of-experiments projects (see Figure 11).

The mode is stored in the `.database` file and is restored the next time you load the project.

To switch from full mode to compact mode:

► Choose **View** > **Flow View Mode** > **Compact**.

To switch from compact mode to full mode:

► Choose **View** > **Flow View Mode** > **Full**.

Figure 11 Project view with compact mode activated

Project | Scheduler

	Igate	HaloDose	HaloEnergy	Vds	Lgeff	Xj	Ygox	Tox	Vtgm	Vti	Id	SS	gm
1	0.18	1e13	15	0.05	1.533e-01	1.363e-01	6.357e-04	3.307e-03	0.357	0.288	6.865e-05	79.366	7.674e-05
2				1.5	1.533e-01	1.363e-01	6.357e-04	3.307e-03	×	0.255	4.381e-04	78.769	4.578e-04
3			25	0.05	1.521e-01	1.271e-01	6.357e-04	3.307e-03	0.384	0.317	6.776e-05	79.861	7.781e-05
4				1.5	1.521e-01	1.271e-01	6.357e-04	3.307e-03	×	0.284	4.243e-04	79.006	4.572e-04
5		1e12	15	0.05	1.506e-01	1.373e-01	6.357e-04	3.307e-03	0.301	0.238	7.410e-05	77.270	8.286e-05
6				1.5	1.506e-01	1.373e-01	6.357e-04	3.307e-03	×	0.199	4.782e-04	76.941	4.718e-04
7	0.13	1e13	15	0.05	1.029e-01	1.363e-01	6.346e-04	3.388e-03	0.366	0.297	8.929e-05	82.605	1.027e-04
8				1.5	1.029e-01	1.363e-01	6.346e-04	3.388e-03	×	0.220	5.183e-04	83.532	5.145e-04
9			25	0.05	1.022e-01	1.271e-01	6.346e-04	3.388e-03	0.382	0.315	8.947e-05	82.612	1.058e-04
10				1.5	1.022e-01	1.271e-01	6.346e-04	3.388e-03	×	0.234	5.123e-04	82.579	5.151e-04
11		1e12	15	0.05	1.009e-01	1.373e-01	6.346e-04	3.388e-03	0.272	0.204	9.869e-05	82.191	1.114e-04
12				1.5	1.009e-01	1.373e-01	6.346e-04	3.388e-03	×	0.109	5.864e-04	86.280	5.268e-04

Customizing the View of the Current Project

You can customize the view of the current project by showing or hiding the following flow elements manually:

- Tool instance
- Default tool step of a tool instance
- Parameter step of a tool instance
- Variable

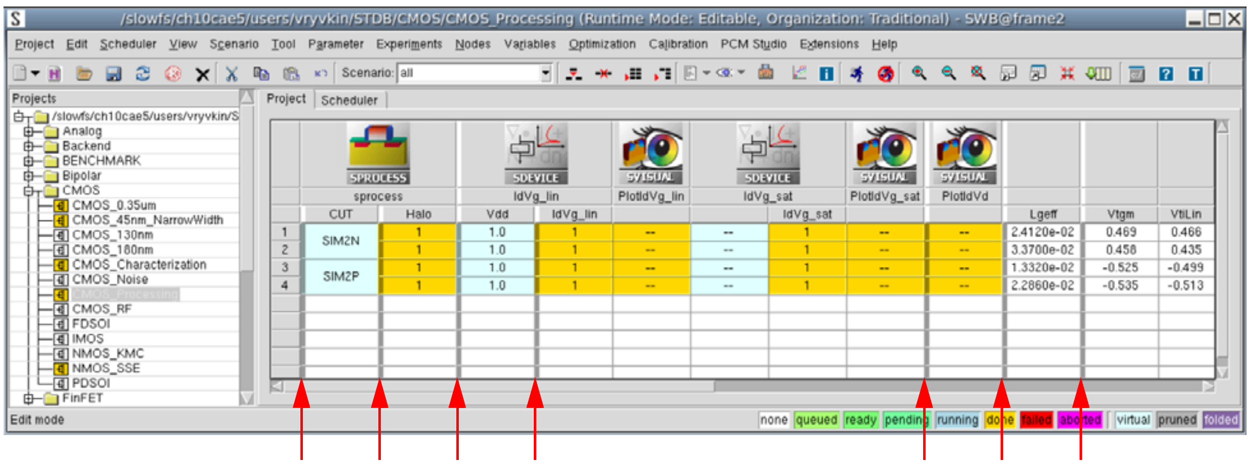
To hide a flow element, for example, a parameter step:

► Select a cell separation line to the right of the parameter name cell and drag the line to the left until the cell disappears. Release the mouse button.

Alternatively, choose **Parameter** > **Hide**.

This operation hides the parameter step. To remind you that a flow element is hidden, Sentaurus Workbench shows the corresponding cell separator line in bold (see Figure 12).

Figure 12 Project with hidden tools, parameters, and variables (red arrows indicate hidden flow elements)



To redisplay a hidden parameter step:

► Select the bold cell separation line and drag the line to the right. Release the mouse button.

Alternatively, choose **Parameter** > **Show**.

You can show or hide other flow elements in a similar way.

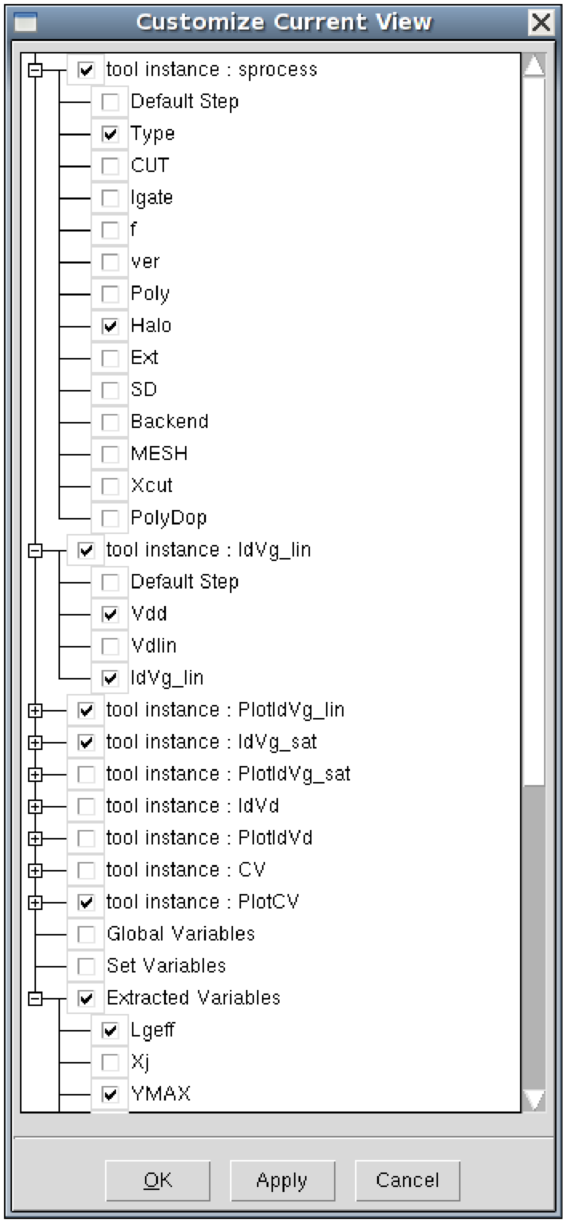
For convenience, you can use the Customize Current View dialog box to show or hide different flow elements in one step (see Figure 13).

To display the Customize Current View dialog box:

► Choose **View** > **Customize Current View**.

All your customizations are stored in the view settings `.database` file and are restored the next time you load the project.

Figure 13 Customize Current View dialog box



Changing the Font of the Project View

You can change the font attributes of the project view for the currently open project.

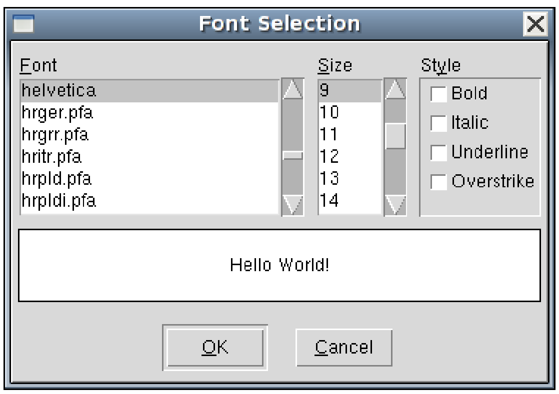
To change the font of the project view:

1. Choose **View** > **Table Options** > **Change Table Font**.

   The Change Font dialog box opens.

2. Select one of the following options:

   - **Apply System Default Font** specifies that Sentaurus Workbench automatically detects the optimal font to use.
   - **Choose Font from Dialog** displays the Font Selection dialog box where you can select the font attributes that are available for the operating system.

3. Click **OK**.

The next time the project is loaded, the font settings are applied.

To apply a particular font to all projects:

1. Choose **Edit** > **Preferences**.
2. In the SWB Preferences dialog box, expand **Table** > **Font**.
3. Configure the font attributes in a similar way as configuring the font of the Family Tree.
4. Click **Apply**.

Note:

Font attributes set up in the preferences apply to both the Family Tree and the projects browser.

The font attributes of the Family Tree are stored in the `.database` file and these attributes overwrite font settings in the preferences.

Changing the Application Font

Sentaurus Workbench uses the default system font as the default application font except for the project view, that is, the main menus, projects browser, and dialog boxes.

You can change the attributes of the default application font in the user-level X configuration file `.Xdefaults` using the standard X logical font description with the `swb*font` name. For example:

```
swb*font: -adobe-helvetica-medium-r-n*-14-*-75-75-*-iso10646-1
```

To change the default application font used to preview node files in the Node Explorer and to view output files, specify a fixed font with the `swb*fixedfont` name. For example:

```
swb*fixedfont:  -*-courier-medium-r-*-*-20-*-*-*-*-*-*-*
```

Configuring the Column Width and Row Height

By default, Sentaurus Workbench automatically calculates the column widths and the row heights to adjust the project data. You can customize the width of all columns and the height of all rows in the project table by manually changing their size using mouse operations.

Sentaurus Workbench stores all the column widths and row heights in the view settings `.database` file and applies these sizes the next time you load the project.

To restore the default column and row sizes at any time:

► Choose **View** > **Restore Default Cell Size**, or press Ctrl+7.

Magnifying the Project View

You can change the magnification of the project view. When a project has a large Family Tree, it can be useful to zoom out of the project view.

Sentaurus Workbench will display values in a smaller font size, which allows you to reduce the cell size and to see a bigger part of the parameterization table at one time. Zooming in will result in the opposite effect. The font name and other font attributes are retained without change (see Changing the Font of the Project View on page 58).

To zoom in to the project view:

► Choose **View** > **Zoom In**, press Ctrl+Plus sign (+), or click the **Zoom In** button.

To zoom out of the project view:

► Choose **View** > **Zoom Out**, press Ctrl+Minus sign (-), or click the **Zoom Out** button.

To switch the zoom off from a previously zoomed part of the project view:

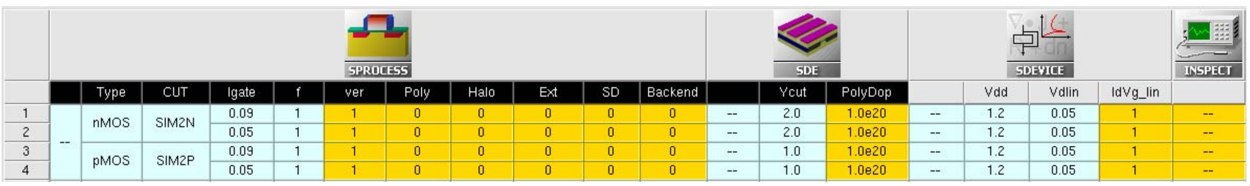
► Choose **View** > **Zoom Off**, press Ctrl+0, or click the **Zoom Off** button.

Freezing Columns and Rows

When a project has many tools and parametric steps, you might want to *freeze* a certain part of the Family Tree and keep it visible on your screen when scrolling through the rest of the Family Tree to the right.

For example, imagine a simulation setup that starts with a process simulation and remeshing followed by multiple device tests. You might want to configure the project view so that the process simulation and remeshing part remains visible on-screen when you scroll through the device tests part (see Figure 14).

Figure 14 Project with frozen columns (indicated by cells with black background)



		Type	CUT	Igate	f	ver	Poly	Halo	Ext	SD	Backend		Ycut	PolyDop		Vdd	Vdlin	IdVg_lin	
1	--	nMOS	SIM2N	0.09	1	1	0	0	0	0	0	--	2.0	1.0e20	--	1.2	0.05	1	--
2				0.05	1	1	0	0	0	0	0	--	2.0	1.0e20	--	1.2	0.05	1	--
3		pMOS	SIM2P	0.09	1	1	0	0	0	0	0	--	1.0	1.0e20	--	1.2	0.05	1	--
4				0.05	1	1	0	0	0	0	0	--	1.0	1.0e20	--	1.2	0.05	1	--

To freeze columns or rows:

1. Select the columns in the parameter row or tool row.
2. Choose **View** > **Freeze Rows/Columns**.

Sentaurus Workbench freezes the selected columns, so they remain visible on-screen when you scroll through the Family Tree. Frozen columns are shown in black, which identifies them the next time you load the project.

To unfreeze columns or rows:

► Choose **View** > **Unfreeze Rows/Columns**.

In addition, you can select an arbitrary set of nodes and freeze it. As a result, Sentaurus Workbench will determine the rectangular area of the Family Tree and freeze the area.

Copying Data to Other Tools

You can easily copy data from the table on the **Project** tab to other tools. Spreadsheet applications support direct copy-and-paste operations of tabbed data.

Main Sections of Project View

The main sections of the project view are:

- Family Tree
- Experimental Plan
- Parameter Values
- Variable Values

Family Tree

The Family Tree shows the simulation flow, which is the backbone of all projects. It defines the sequence of tools and parameters involved in the simulations. Each parameter belongs to a tool instance, and experiments are arranged vertically. The table cells are either real or virtual nodes associated with individual simulations.

To show or hide the Family Tree:

► Choose **View** > **Tree Options** > **Show Tree**, or press the F1 key.

Figure 15 Family Tree with horizontal orientation

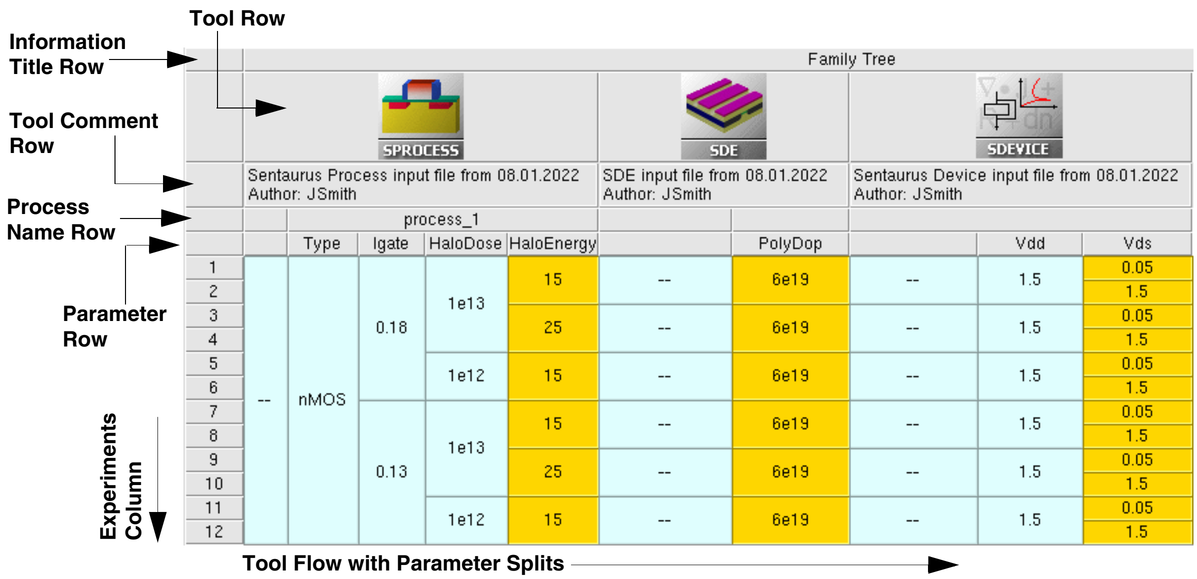
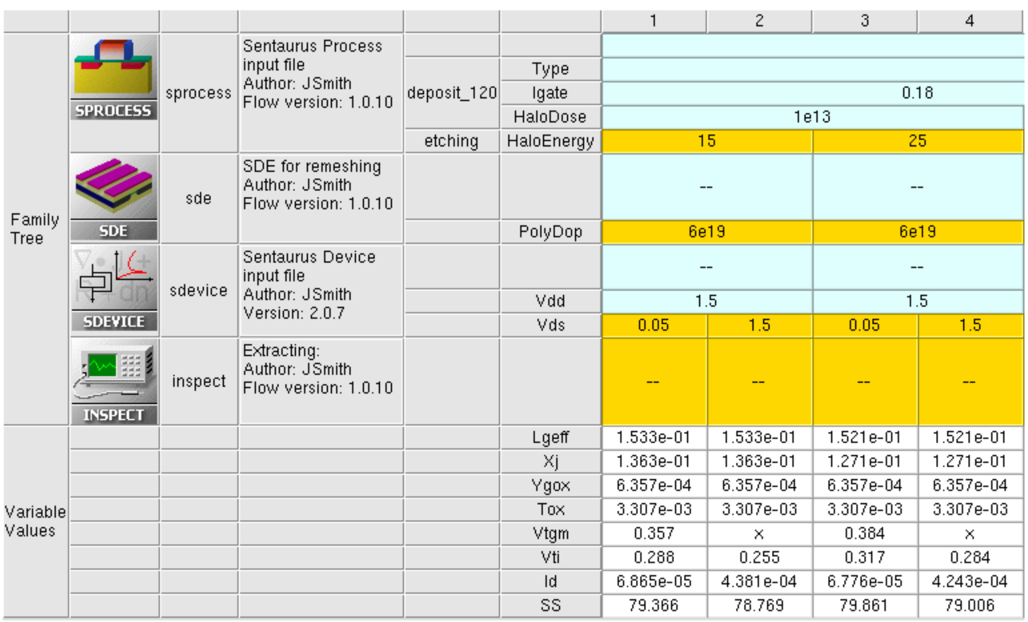


Figure 15 shows different parts of the Family Tree including:

Information title row	Shows the titles of the main parts of project view such as Family Tree and Variable Values.
Tool row	Shows the icons of all tools of the project.
Tool label row (hidden in Figure 15)	Shows all tool labels directly below the tool row. In the SWB Preferences dialog box, expand **Default View Options** and set **Display Tool Labels** to true, and reload the project.
Tool comment row	Shows comments about tools in the project.
Process name row	Shows the process names for the parameters.
Parameter row	Shows all parameters of the project.
Experiments column	Shows all numbered experiments.

When the project view is oriented vertically, rows become columns and vice versa, so the project view looks like Figure 16.

Figure 16 Family Tree with vertical orientation



You can change the Family Tree view to display information about a node or from the node status files. Use the commands available from **View** > **Tree Options** and **View** > **Table Options** (see Table 27 on page 311).

Note:

To change views sequentially, click the  button.

Specifying a Tool Comment

You can specify comments for any tool in the project flow. The comments are displayed in the tool comment row. Sentaurus Workbench displays this row when at least one tool contains a comment. A comment is arbitrary multiline text.

To specify a tool comment:

1. Choose **View** > **Table Options** > **Show Comments**.

   A text box appears below each tool icon in the tool flow.

2. Type the comment.

Sentaurus Workbench saves tool comments in the project directory in the `gcomments.dat` file and displays the comments the next time you load the project.

Specifying a Parameter Process Name

You can specify a process name for each parameter of all process tools in the tool flow. This allows process engineers who often involve many process variations and splits in process simulations to conveniently group process parameters. A process name is an arbitrary identifier that can reflect certain process steps, splits, and so on.

When you create a Sentaurus Workbench parameter for any process tool, you can also specify the process name to which this parameter belongs. The process name as well as the parameter name can be changed later in the Parameter Properties dialog box.

Process names are displayed in a separate row immediately above the parameter row.

To specify a parameter process name:

► Choose **View** > **Table Options** > **Show Parameter Process Names**.

Sentaurus Workbench saves the specified process names in the core project files and displays them the next time you load the project.

Note:

Process names are supported for process simulators and process-aware tools such as Sentaurus Process, Sentaurus Structure Editor, Sentaurus Topography 3D, and Taurus™ TSUPREM-4™.

Showing the Tool Labels

In addition to tool icons, you can also display the tool label as rollover text. This feature is switched off by default.

To switch on the feature:

► Choose V**iew** > **Tree Options** > **Hinting Tool Labels**.

In addition, you can show the tool labels permanently in the tool label row.

To show the tool label row permanently:

► Choose **View** > **Table Options** > **Show Tool Labels**.

Experimental Plan

The Experimental Plan view provides a way to view parametric combinations. It is used for viewing purposes only. The header rows show all parameters and their values.

For each experiment, all its parameter values are also shown in the Experimental Plan columns in blue, under their corresponding values. This feature can be useful to observe certain patterns in the variation of parameters.

To switch on or off the Experimental Plan view:

► Choose **View** > **Tree Options** > **Show Experimental Plan**, or press the F2 key.

Parameter Values

The Parameter Values view contains the tools and their parameters on separate header rows. It is used for viewing purposes only. The parameter value is shown for each experiment and for each parameter.

To switch on or off the Parameter Values view:

► Choose **View** > **Tree Options** > **Show Parameters**, or press the F3 key.

Variable Values

The Variable Values view is used for viewing, editing, deleting, and adding variables. By default, it shows all variable types (see Variable Values on page 64).

To switch on or off the Variable Values view:

► Choose **View** > **Tree Options** > **Show Variables**, or press the F4 key.

You can hide certain variable types using the Customize Current View dialog box (see Customizing the View of the Current Project on page 55).

Nodes

Nodes are colored according to their status and are labeled with unique numbers called *node keys*. All nodes in the Family Tree can be one of the following:

- *Real nodes* correspond to real simulation phases. For example, if a tool has no parameters, all nodes of this tool are real nodes.

  Real nodes are colored according to the execution status of the corresponding simulation job.

- If a tool has parameters, the parameters introduce intermediate nodes, which do not usually correspond to real simulation phases and, therefore, do not hold any results. These intermediate nodes are *virtual nodes*.

  Virtual nodes are either light blue or white depending on whether you have chosen the option **View** > **Tree Options** > **Check Virtual Nodes**.

Note:

Tools with split capabilities such as Sentaurus Process, Sentaurus Structure Editor, and Taurus TSUPREM-4 can create real intermediate nodes (see Split Points on page 169).

If the **View** > **Tree Options** > **Check Virtual Nodes** option is switched on, Sentaurus Workbench shows the status of intermediate nodes as follows:

- Real intermediate nodes are colored according to their status.
- Virtual intermediate nodes are light blue.

The definitions of the different node colors are shown in the status bar of the main window (see Configuring Node Colors).

For more information about a simulation, choose **Nodes** > **Edit Properties**.

Configuring Node Colors

Nodes are colored according to their status. The default color scheme for nodes is displayed in the status bar (see Figure 17).

Figure 17 Default node colors as shown in the status bar

Sentaurus Workbench recognizes Tk symbolic color names or hexadecimal 8-bit RGB values. For details, go to https://www.tcl.tk/man/tcl8.6/TkCmd/colors.html.

Table 4 RGB values for the default colors of nodes

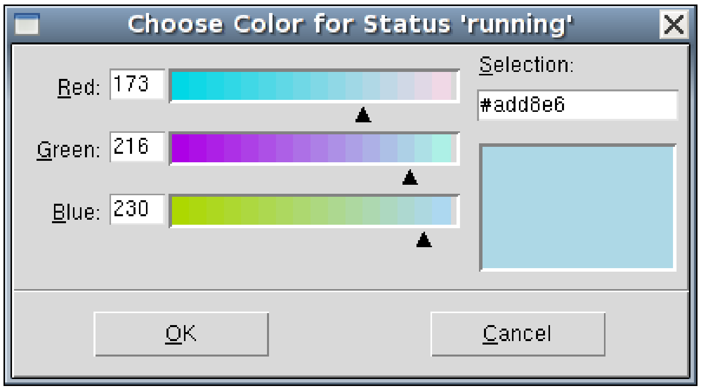
Node status	Hexadecimal 8-bit RGB value
none	#FFFFFF
queued	#ACFF75
ready	#7CFF75
pending	#75FFA0
running	#ADD8E6
done	#FFD700
failed	#FF0000
aborted	#FF00FF
virtual	#E0FFFF
pruned	#B3B3B3
folded	#8465A5

To change the standard color scheme for better differentiation of colors on-screen:

1. Choose **Edit** > **Preferences**.
2. In the SWB Preferences dialog box, expand **Table** > **Node Status Color**.
3. Change the colors in one of the following ways:
   - Specify the hexadecimal RGB value manually.
   - Select the color from the Choose Color for Status dialog box, and click **OK** (see Figure 18).
4. Click **Apply**.

New colors are applied immediately after reloading a project, and they apply to all Sentaurus Workbench projects.

Figure 18 Choose Color for Status dialog box for changing the color of a node status



Selecting Nodes With Mouse and Keyboard Operations

You can select nodes as for any spreadsheet application. You can make a typical rectangular node selection using mouse operations.

Alternatively, to select a large rectangular region where scrolling the Family Tree is needed:

1. Click the upper-left node of the required rectangular region.
2. Scroll the Family Tree until the lower-right node of the rectangular selection appears.
3. Click this node while holding the Shift key.

Clicking cells with experiment numbers results in selecting all the nodes belonging to these experiments. Clicking tool or parameter cells while holding the Shift key allows you to select all nodes belonging to the given tools or parameters.

When combining different selection techniques with the Ctrl key, you can make complex selections containing multiple regions.

Viewing the Output Files of Nodes

Each real node can have several output files. You can view the contents of these files using Sentaurus Visual (the default visualization tool in Sentaurus Workbench), Inspect, or the text editor.

To view all output files of one or more nodes:

1. Select the nodes on the **Project** tab.
2. Choose **Nodes** > **Quick Visualize**, or click the **Quick Visualize** toolbar button.

All of the available output files for the selected nodes open in the default visualizer, Sentaurus Visual. If no output files correspond to the selected nodes, Sentaurus Workbench launches the default visualizer without any files.

To view all output files of one or more nodes with a specific visualization tool:

1. Select the nodes on the **Project** tab.
2. Click the **Visualize** toolbar button, and select the required visualizer.

   All visualization tools available for the selected nodes are listed.

   Alternatively, to make all visualization tools available, choose **Nodes** > **Visualize**.

   If no output files correspond to a particular node, the visualizer will be launched empty.

To view the output files of nodes belonging to different projects:

1. Select multiple projects in the projects browser.
2. Select the required nodes of the currently open project on the **Project** tab.

Viewing the output files of nodes displays those files belonging to the current project as well as the output files belonging to nodes with the same node numbers in other selected projects (if they exist).

Note:

> The currently open project must be one of the selected projects in the projects browser.

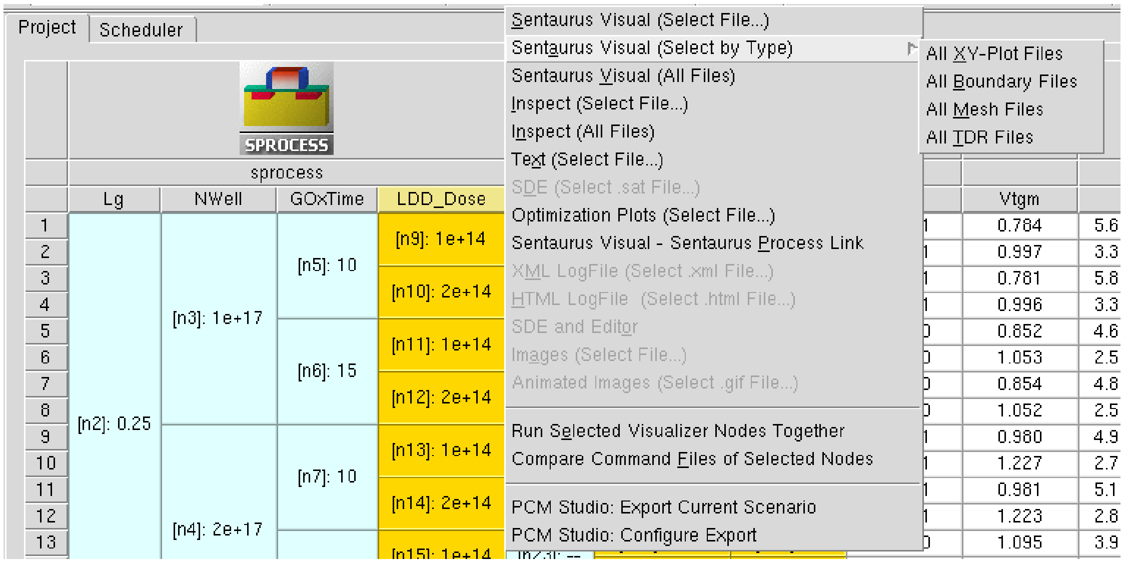
The visualization tools, the file patterns to visualize, and the maximum number of files are configured in the tool database. By default, Sentaurus Workbench configures several viewers as shown in Figure 19.

The visualization tools appear more than once in the following modes:

- **Select File**: Sentaurus Workbench prompts you to select files for visualization.
- **Select by Type**: Sentaurus Workbench allows you to visualize files according to type. This mode is available for Sentaurus Visual only. If the number of such files exceeds the maximum number specified in the tool database for this viewer, Sentaurus Workbench prompts you to select files as in the **Select File** mode.
- **All Files**: All files of the selected nodes are visualized without prompting. If the number of such files exceeds the maximum number specified in the tool database for this viewer,

Sentaurus Workbench prompts you to select files as in the **Select File** mode. You will be also prompted to select the target Sentaurus Visual instance if you have one or more running Sentaurus Visual instances launched from the current Sentaurus Workbench session (see Viewing the Output Files of Nodes in Sentaurus Visual on page 70).

Figure 19 Output files visualization menu



Sentaurus Workbench offers the following predefined file filters for the **Select by Type** mode:

- All XY-Plot Files (`*.plt`, `*.plx`)
- All Boundary Files (`*bnd.tdr`)
- All Mesh Files (`*msh.tdr`)
- All TDR Files (`*.tdr`)

Each file filter defines a list of file patterns separated by space. Additional file filters can be created in the Sentaurus Visual Visualization dialog box (see Figure 20). You also can change the patterns of the existing file filters in the preferences (expand **Visualization** > **File Filters**).

You can define any viewer as the default visualization tool in the tool database by using the `WB_tool(default,visualizer)` variable. In Sentaurus Workbench, the default viewer is Sentaurus Visual.

You can define any viewer as the default visualization tool for unrecognized files in the tool database by using the `WB_tool(unrecognized_files,visualizer)` variable.

Viewing the Output Files of Nodes in Sentaurus Visual

Sentaurus Visual is the default visualization tool in Sentaurus Workbench. The comprehensive integration of Sentaurus Workbench and Sentaurus Visual provides special capabilities that are not available in Inspect.

You can view the output files in a new instance of Sentaurus Visual or in an already running instance of Sentaurus Visual.

Sentaurus Workbench detects the Sentaurus Visual instances that you launched from the current instance of Sentaurus Workbench, and it displays the Sentaurus Visual Visualization dialog box (see Figure 20).

To visualize the output files of nodes in Sentaurus Visual:

1. In the Sentaurus Visual Visualization dialog box, select files in the left pane.

   See Selecting Files Using Filters on page 71.

2. Choose the instance of Sentaurus Visual to use in the right pane.

   Already running Sentaurus Visual instances are noted as SWB_1, SWB_2, and so on. To identify the corresponding instance of Sentaurus Visual, check the title of the main window of Sentaurus Visual, which displays the same identification.

   Note:
   Sentaurus Workbench recognizes only those instances of Sentaurus Visual that were launched from itself.

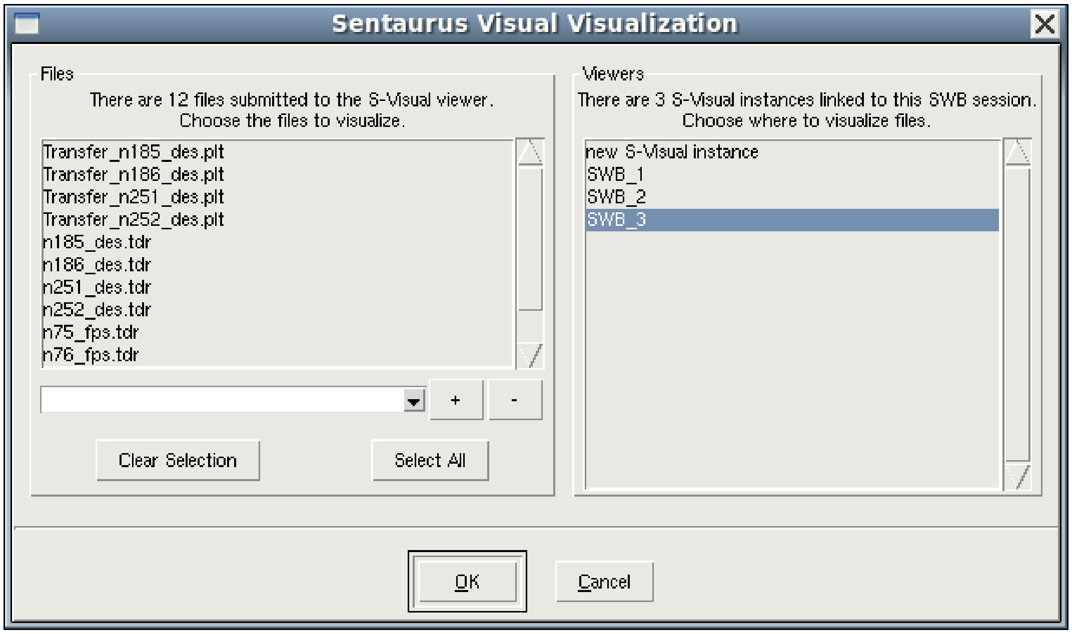
3. Depending on the number of already running Sentaurus Visual instances launched from the current Sentaurus Workbench session, you can choose to visualize selected files in either the last-used Sentaurus Visual instance (the one you used last time to visualize files), or the last-launched Sentaurus Visual instance, or a new Sentaurus Visual instance.

   Sentaurus Workbench selects the correct Sentaurus Visual instance according to the preferences (expand **Visualization** > **Default S-Visual Instance**, whose value can be **Last Used**, **Last Created**, or **New Instance**).

   The default behavior is to visualize the selected files in the last-used Sentaurus Visual instance.

4. Click **OK**.

Figure 20 Sentaurus Visual Visualization dialog box



Selecting Files Using Filters

The number of files that can be visualized can be very large. If you apply file filters using the text box below the left pane, this will reduce the list, displaying only the files you want.

Each file filter specifies one or more file patterns, and only files matching at least one pattern of the selected file filter is displayed.

The following predefined file filters are available from the list of the text box:

- All XY-Plot Files (`*.plt`, `*.plx`)
- All Boundary Files (`*bnd.tdr`)
- All Mesh Files (`*msh.tdr`)
- All TDR Files (*`.tdr`)
- S-Process Command File (`*.cmd`)

Alternatively, you can enter your own space-separated file patterns in the text box, and Sentaurus Workbench automatically updates the list of matching files.

If you want to reuse a file filter, click the **+** button next to the text box and, in the dialog box that is displayed, enter the name of the file filter. Next time, your file filter will appear in the list of the text box and in the **Select by Type** menu (see Viewing the Output Files of Nodes on page 67). Similarly, you can remove any of the file filters by clicking the **-** button. You also can change the patterns of the existing file filters in the preferences (expand **Visualization** > **File Filters**).

Note:

The initial list of files in the left pane of the Sentaurus Visual Visualization dialog box is based on the file patterns defined for the `"svisual"` viewer in the tool database. By default, this list contains the output files of the selected nodes. File filters apply to the initial list of files of the selected nodes.

Viewing Visualizer Nodes Simultaneously

You can view one or more visualizer nodes simultaneously. Visualizer nodes are those belonging to either Sentaurus Visual or Inspect tool instances.

Caution:

You cannot mix Sentaurus Visual and Inspect nodes in a selection.

Sentaurus Visual has a Tcl mode and Python mode. You cannot mix Sentaurus Visual nodes in Tcl and Python modes.

To view visualizer nodes simultaneously:

1. Select one or more Sentaurus Visual or Inspect nodes on the **Project** tab.
2. Choose **Nodes** > **Visualize** > **Run Selected Visualizer Nodes Together**, or click the **Run Selected Visualizer Nodes Together** toolbar button.

Sentaurus Workbench merges the input command files of the selected nodes into one command file and runs the merged command file in the corresponding visualization tool (Sentaurus Visual or Inspect).

By default, Sentaurus Workbench does not recreate the Sentaurus Visual or Inspect node command files from the master Sentaurus Visual or Inspect file, if the node files already exist. If you make a change to the Sentaurus Visual or Inspect master file and want that change to have an effect when choosing **Visualize** > **Run Selected Visualizer Nodes Together**, then you must preprocess or run the Sentaurus Visual or Inspect nodes. Alternatively, you can force Sentaurus Workbench to preprocess the selected nodes before merging them by setting **Visualization** > **Run Selected Visualizer Nodes Together** > **Always Preprocess Nodes** to **Yes** in the preferences. In this case, Sentaurus Workbench will always preprocess nodes before merging even though the node files already exist.

Note:

Forced preprocessing of selected nodes might slow down visualization of large design-of-experiments projects.

The visualization tool always launches in interactive mode on the local host. You can view selected visualizer nodes simultaneously at any time regardless of the project status and the node dependencies. Sentaurus Workbench splits the visualization output into node-specific parts, extracts the corresponding simulation results, and updates the extracted results in the Family Tree.

You can use this capability to extract and visualize the intermediate results of long-running simulations. You can analyze the extracted data before a simulation is fully completed. For example, you can see the intermediate results of a running 3D device simulation by visualizing the next extraction Sentaurus Visual node. Doing this for multiple extraction nodes allows you to compare the intermediate results.

Sentaurus Workbench injects the following settings at the beginning of the merged command file of Sentaurus Visual:

- In Tcl mode:

  ```
  set runVisualizerNodesTogether 1
  set runInspectNodesTogether 1
  set extractNodeResults 0
  ```

- In Python mode:

  ```
  runVisualizerNodesTogether = True
  runInspectNodesTogether = True
  extractNodeResults = False
  ```

You can use these settings to implement different behavior of the same visualization script being run in batch mode and in interactive mode. The first two settings allow you to detect whether the visualization tool has been launched with the **Run Selected Visualizer Nodes Together** toolbar button in Sentaurus Visual and Inspect, respectively. For example, you can include the following logic in your Sentaurus Visual script in Tcl mode:

```
if { [info exists runVisualizerNodesTogether] } {
   # --------- Plotting ----------
   ...
} else {
   #--------- Extraction ---------
   ...
}
```

and in Python mode:

```
if 'runVisualizerNodesTogether' in globals():
   # --------- Plotting ----------
   ...
else:
   #--------- Extraction ---------
   ...
```

Usually, you do not want to replace the extracted results of the Sentaurus Visual batch execution with the results of the interactive run when clicking **Run Selected Visualizer Nodes Together**. That is why Sentaurus Workbench does not extract the results from the node output file by default. However, if you still want to extract results after clicking **Run Selected Visualizer Nodes Together**, then you can set the corresponding setting in your visualization script as follows:

- In Tcl mode: `set extractNodeResults 1`
- In Python mode: `extractNodeResults = True`

A complex evaluation on half-finished data can fail, for example, to determine the maximum of a curve, if the curve does not contain a maximum yet. It can be complicated to determine whether a simulation has finished based only on the data. Therefore, in critical parts of the Sentaurus Visual and Inspect input command files, you can specify a node-readiness check to prevent this issue.

For example, in Tcl mode:

```
set status @[gproject::GetNodeStatus @node|sdevice@]@
if { $status == "done" } {
   # Sentaurus Device node is completed, so proceed with data evaluation
}
```

and in Python mode:

```
status = "@[gproject::GetNodeStatus @node|sdevice@]@"
if status == "done":
   # Sentaurus Device node is completed, so proceed with data evaluation
```

Comparing Command Files of Nodes

You can compare the content of command files of selected nodes using a comparison application.

The default comparison application is *tkdiff*. You can change this in the preferences (choose **Edit** > **Preferences** and, in the SWB Preferences dialog box, expand **Utilities** > **Diff Tool**).

To compare command files:

1. Select one or two nodes on the **Project** tab.
2. Choose **Nodes** > **Visualize** > **Compare Command Files of Selected Nodes**.

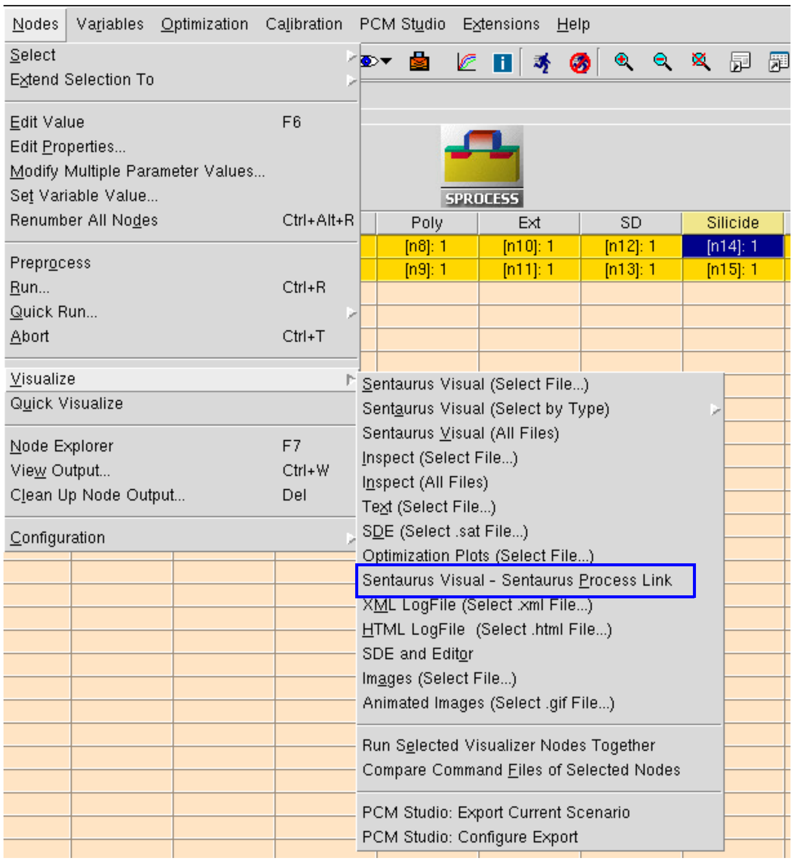
If one node is selected, then Sentaurus Workbench compares its command file to the master command file of the corresponding tool. If two nodes are selected, their command files are compared.

Running Sentaurus Process Nodes Interactively

From Sentaurus Workbench, you can open the interface between Sentaurus Visual and Sentaurus Process to interactively run the process flow of a selected Sentaurus Process node.

To open the interface:

1. Select the required Sentaurus Process node.
2. Choose **Nodes** > **Visualize** > **Sentaurus Visual - Sentaurus Process Link**.



Sentaurus Workbench launches Sentaurus Visual with the already loaded Sentaurus Process flow.

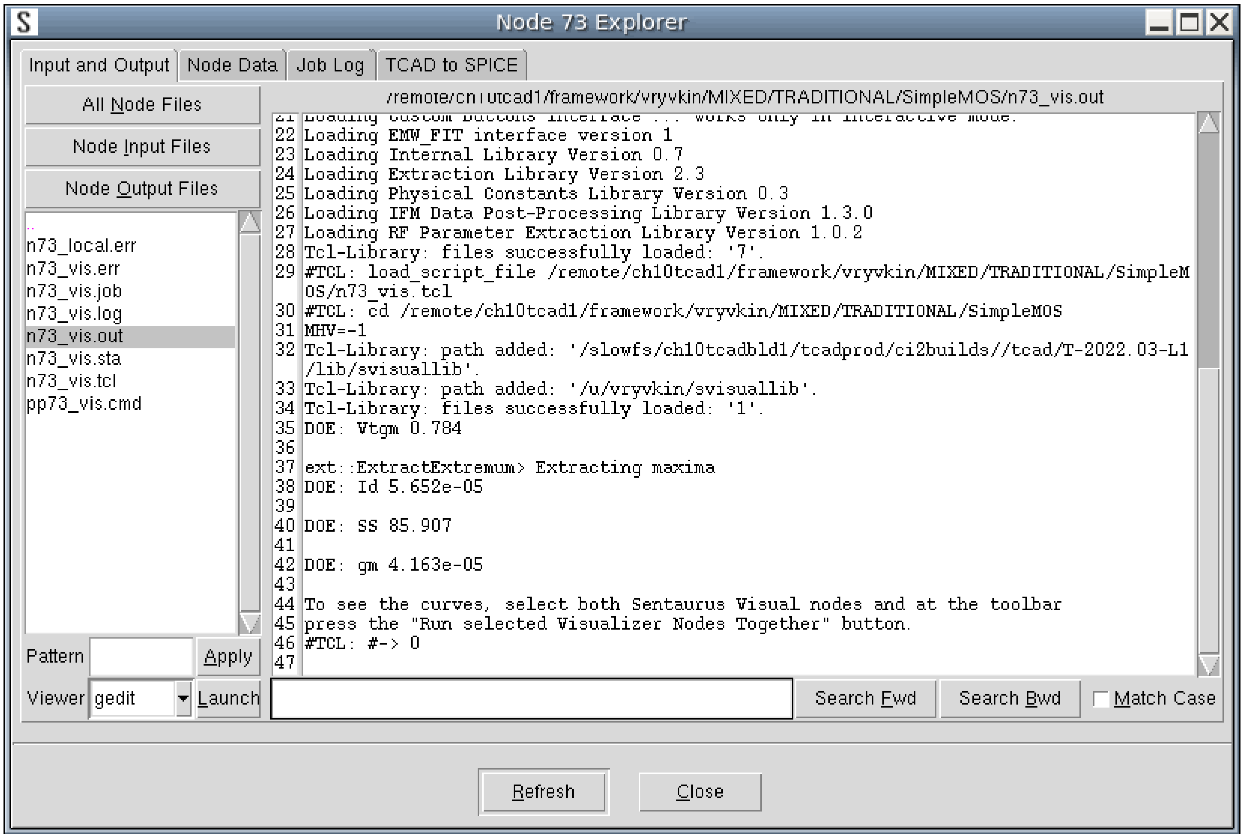
Alternatively, choose **Extensions** > **Run Sentaurus Visual - Sentaurus Process Link** to open the interface directly with an empty process flow. You must load the Sentaurus Process flow manually.

See *Sentaurus™ Process User Guide*, Interface to Sentaurus Visual, for more information about the interface.

Node Explorer

The Node Explorer assembles all node-related data and files in one place, which simplifies navigating through node files, analyzing simulation results, and tracking simulation problems.

Figure 21 Node Explorer showing Input and Output tab with text preview area



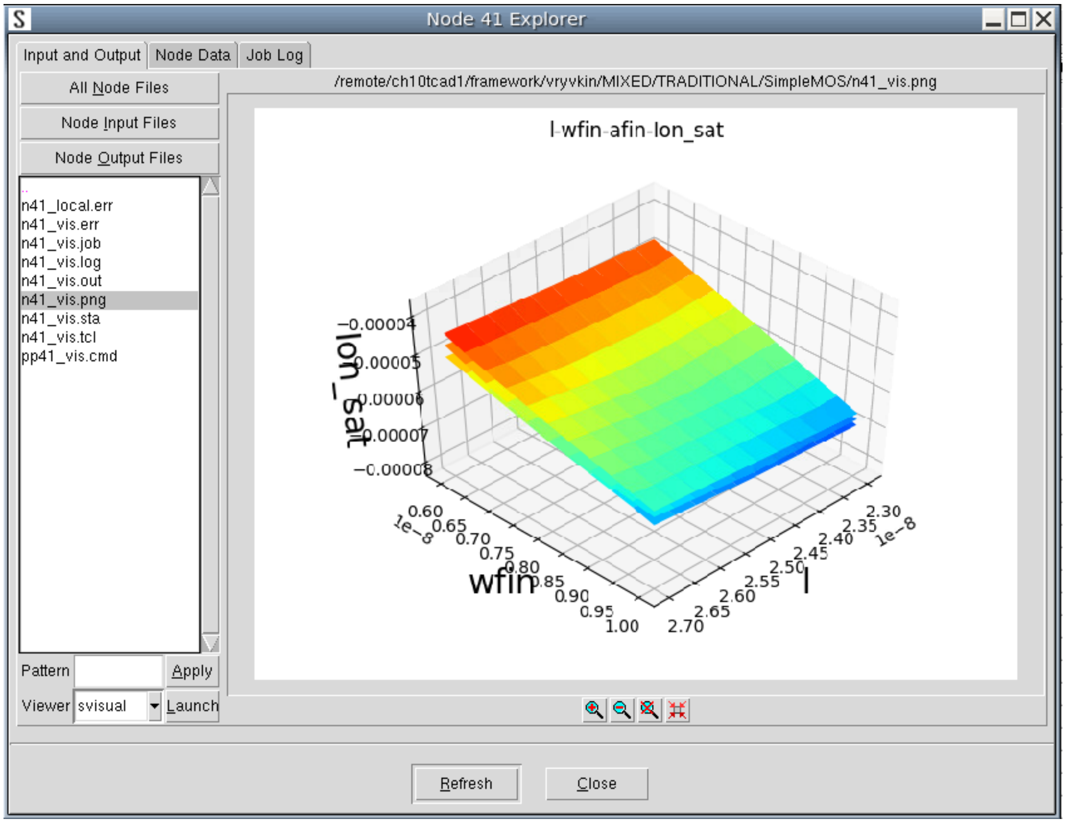
The Node Explorer has the following tabs:

- The **Input and Output** tab allows you to navigate through node files and directories. You can visualize node files in dedicated external viewers.

  You can view text files in the preview area and search for content in both forward and backward directions. By default, the text preview area displays line numbers, and you can mark lines by clicking the line number. However, you can switch off these features in preferences, by expanding **Node Explorer** > **Line Numbering** and **Node Explorer** > **Markable Line**.

  You can view graphic node files in `.png`, `.gif`, and `.jpg` format in the image preview area. Animated graphics are not supported in the image preview area (see Figure 22).

Figure 22 Node Explorer showing Input and Output tab with image preview area



- The **Node Data** tab displays node properties, together with the defined parameters and extracted data (see Figure 23). You can view the properties of the node. You can see the values of Sentaurus Workbench parameters and variables defined or extracted on the current node. By default, Sentaurus Workbench displays the values of parameters and variables defined or extracted on upstream nodes, but you can change this by deselecting the corresponding options.
- The **Job Log** tab displays the log file of the node (see Figure 24). You can use buttons on the left to parse specific parts of the log file. This can help to track the root cause of a job failure.
- The **TCAD to SPICE** tab allows you to export node data to the project database (see Figure 25). This tab is available only for the nodes of TCAD to SPICE tools.

Figure 23 Node Explorer showing Node Data tab

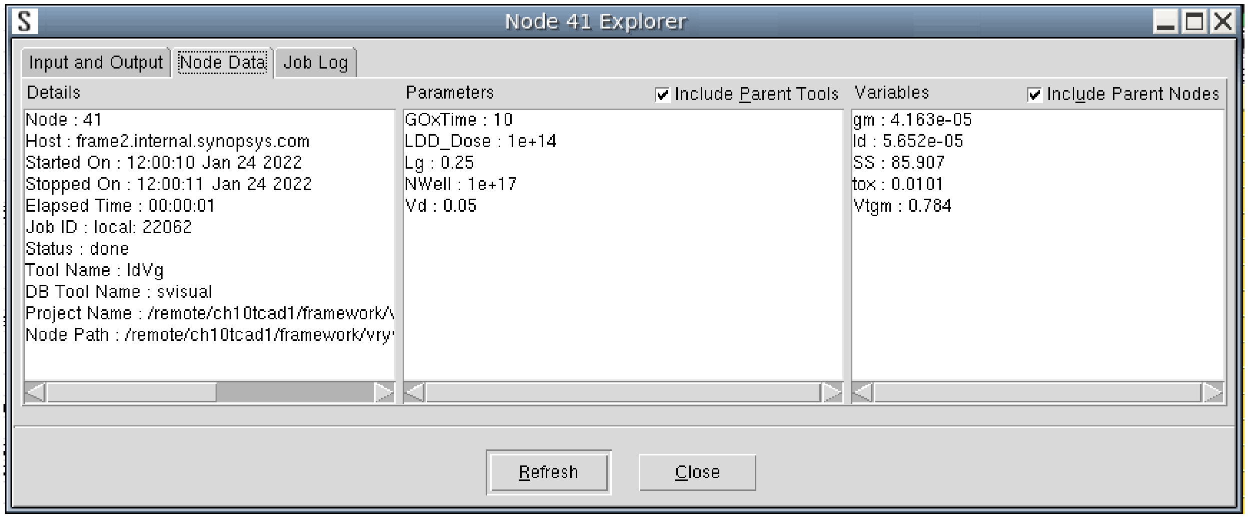


Figure 24 Node Explorer showing Job Log tab

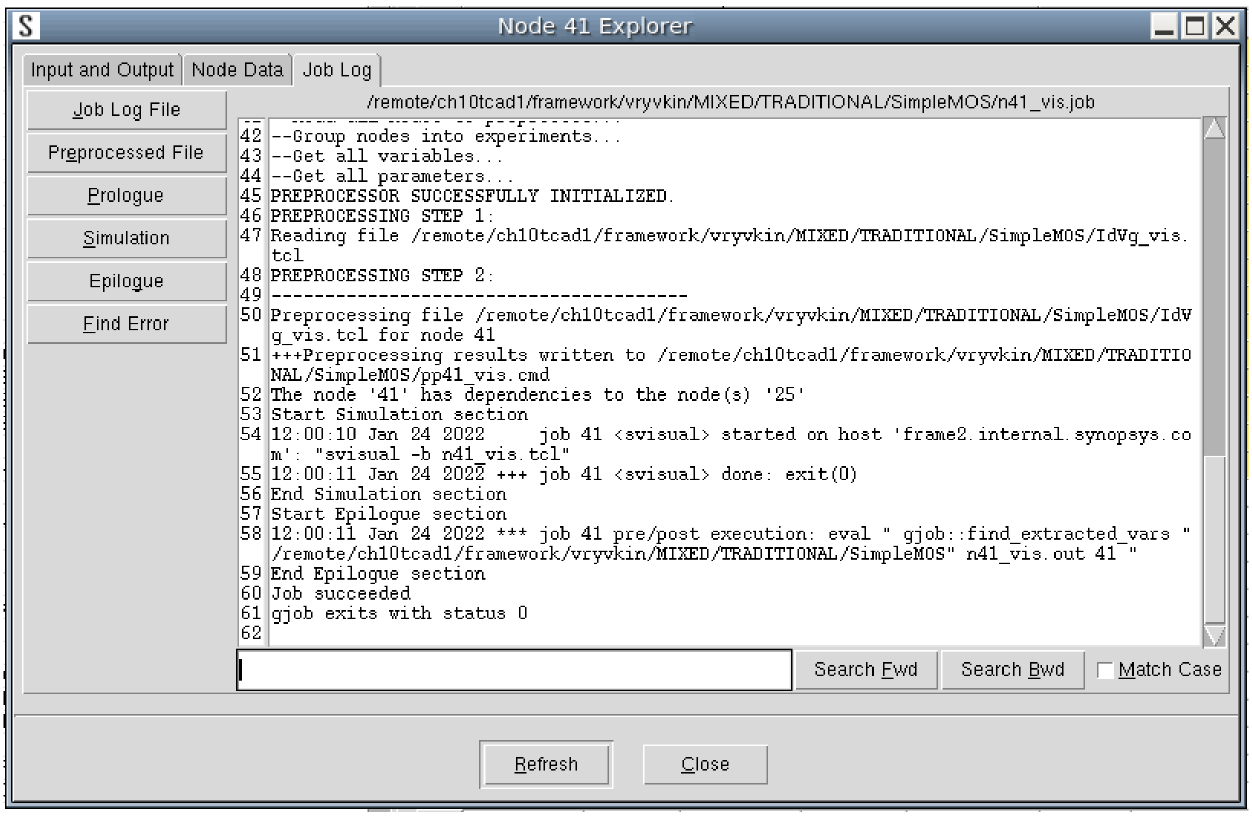


Figure 25 Node Explorer showing TCAD to SPICE tab

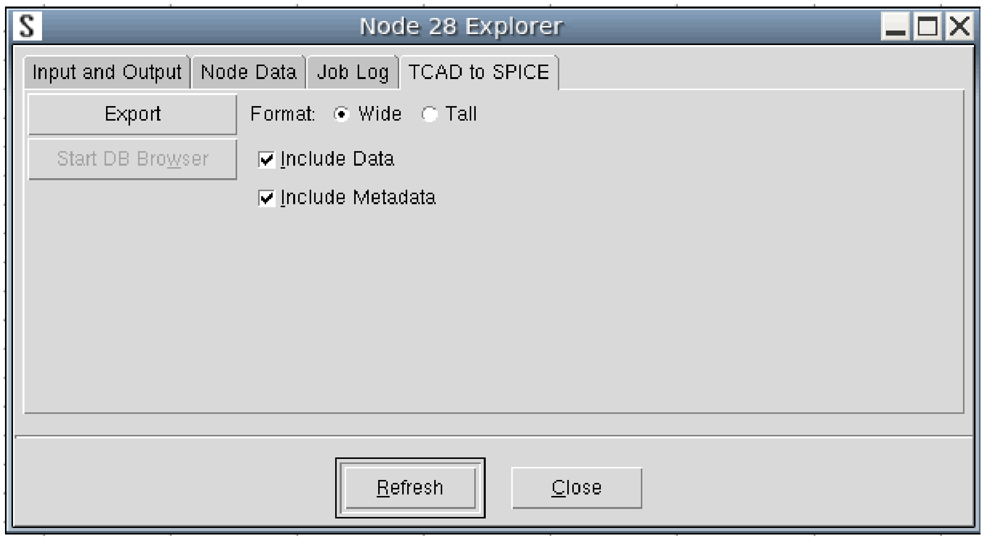


Table 5 Buttons and options in Node Explorer

Button or option	Description
All Node Files	Displays all files that are associated with the current node. To determine these files, tool database file patterns are used.
Node Input Files	Displays the input files of the current mode, which are registered in the tool database.
Node Output Files	Displays the output files of the current node, which are registered in the tool database.
Apply	Applies the file search based on the specified file pattern.
Launch	Every file that is displayed in the upper list box has one or more associated viewers. This button opens the currently selected viewer together with the currently selected file. Files with no associated viewers are considered to be unrecognized files and are visualized with the text viewer.
Search Fwd	Node Explorer looks for the specified pattern in the currently previewed file in a forward direction. Shortcut keys: Enter, Alt+F.
Search Bwd	Node Explorer looks for the specified pattern in the currently previewed file in a backward direction. Shortcut keys: Shift+Enter, Alt+B.
Match Case	Switches case sensitive search on and off. Default: off. Shortcut keys: Alt+M.

Table 5 Buttons and options in Node Explorer (Continued)

Button or option	Description
Include Parent Tools	Specifies whether to show parameters defined on upstream tools.
Include Parent Nodes	Specifies whether to show variables extracted on upstream tools.
Job Log File	Displays the job log file (output of node `gjob` process).
Preprocessed File	Displays the preprocessed node file.
Prologue	Displays the prologue section of the job log file.
Simulation	Displays the simulation section of the job log file.
Epilogue	Displays the epilogue section of the job log file.
Find Error	Assists you to find the reason for a node failure. Sentaurus Workbench analyzes the prologue, simulation, and epilogue sections of the job log file for error messages. In addition, it checks the standard error and standard output files of the simulation for additional error information.
Refresh	Updates the node information.

Opening the Node Explorer

To open the Node Explorer:

1. Select a node.
2. Choose **Nodes** > **Node Explorer**, or press the F7 key.

When the Node Explorer opens, it shows the end of the simulator standard output file. The content of this file is updated dynamically when the node is running.

By default, Sentaurus Workbench displays the Node Explorer in the foreground on top of the Sentaurus Workbench window as a standalone window, which can be minimized. You can change this behavior in preferences, by expanding **Node Explorer** > **Always in Foreground**= Yes to have the Node Explorer always visible.

Exporting Spreadsheets

This section describes how to export spreadsheets for viewing in spreadsheet applications or Inspect.

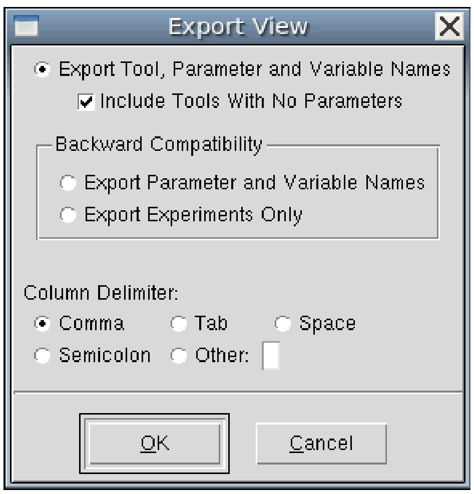
Exporting a Spreadsheet to a Text File

A spreadsheet can be exported as a character-separated value file, which can be loaded into different spreadsheet applications.

To export a spreadsheet to a text file:

1. Choose **Experiments** > **Export** > **Text File**.

   The Export View dialog box opens.



   By default, Sentaurus Workbench adds three rows at the beginning of the exported text file:

   ◦ First row: Tool names

   ◦ Second row: Tool labels

   ◦ Third row: Parameter and variable names

   It allows Sentaurus Workbench to export experiments into a new project. You can also choose to add tools without parameterization into the exported text file.

2. If you plan to import experiments from the text file into older versions of Sentaurus Workbench, select one of the backward compatibility options:
   - Select **Export Parameter and Variable Names** so that Sentaurus Workbench adds parameter and variable names as the first row in the exported file.
   - Select **Export Experiments Only** so that the exported file contains the values of parameters and variables.
3. Select the column delimiter.
4. Click **OK**.

   The File dialog box opens.
5. Enter the file name.
6. Click **Save**.

To open spreadsheet software with the current view:

► Choose **Experiments** > **Export** > **Run Spreadsheet Application**, or click the  button.

This saves the current view to a temporary text file and loads it with a spreadsheet application that is configured in the preferences (expand **Utilities** > **Spreadsheet Application**).

Viewing Log Files

To view the preprocessor and project log files for projects of Sentaurus Workbench:

► Choose **Project** > **Logs** > **Preprocessor** or **Project** > **Logs** > **Project**.

To open an editor with the optimization input file:

► Choose **Optimization** > **View Log**.

Visualizing Response Surfaces

See Visualizing Response Surface Models on page 256.

View Settings for Scheduler

The **Scheduler** tab lists all the running nodes that belong to the selected running projects in the projects browser. The status of nodes is updated based on the refresh frequency setting.

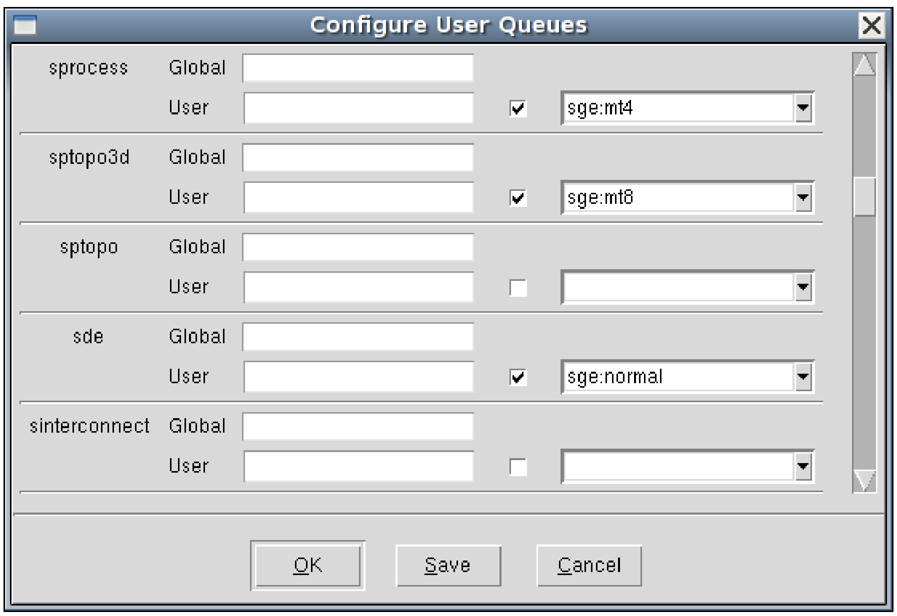
Modifying User-Level Tool Queues

To modify user-level tool queues (see Tool Associations on page 279):

1. Choose **Scheduler** > **Configure Queues** > **User Queues**.

   The Configure User Queues dialog box opens.

2. Modify the tool queue assignments, and enter specific options associated with the Scheduler.



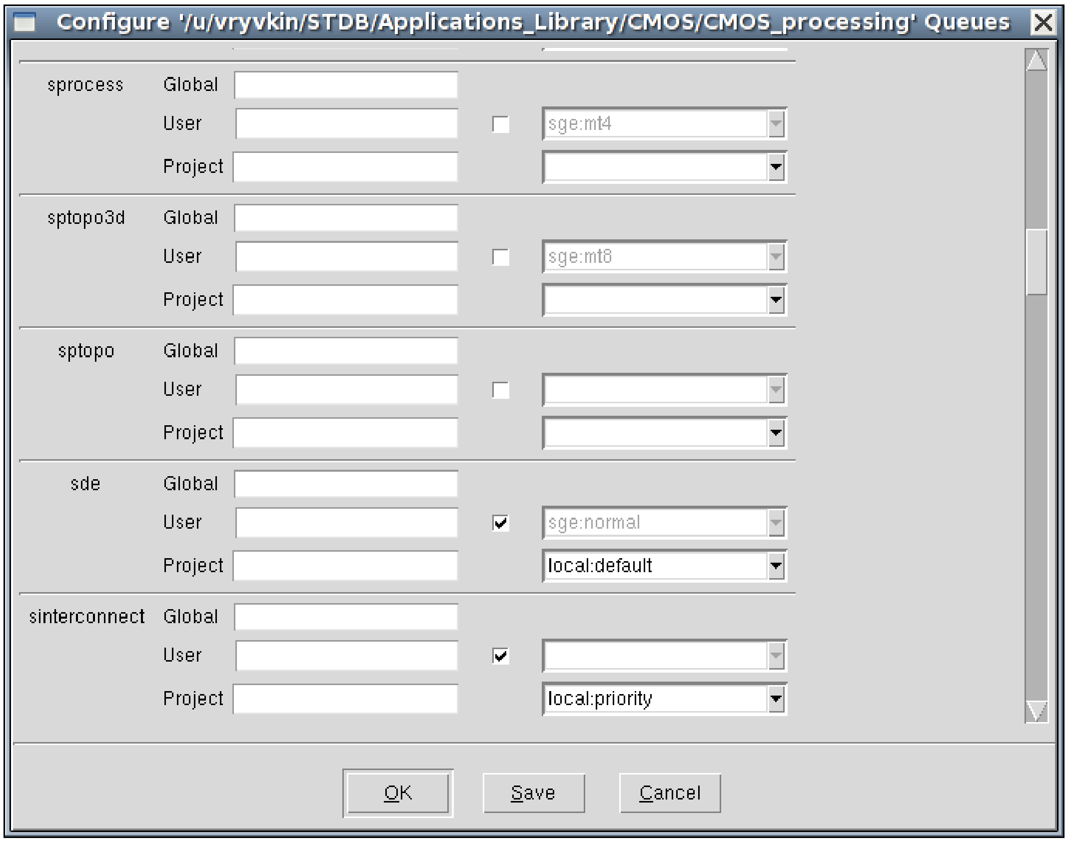
3. Click **OK**.

Modifying Project-Level Tool Queues

To modify project-level tool queues (see Tool Associations on page 279):

1. Choose **Scheduler** > **Configure Queues** > **Project Queues**.

   The Configure Project Queues dialog box opens.



2. Modify the tool queue assignments.

   These assignments apply only to a particular project.

3. Click **OK**.

Editing Queue Files

The Scheduler does not have an interface to add node constraints for user-level or project-level queues. You can only do so by editing the user queue or project queue files manually.

To edit user queue files:

► Choose **Scheduler** > **Configure Queues** > **Edit User Queues**.

To edit project queue files:

► Choose **Scheduler** > **Configure Queues** > **Edit Project Queues**.

4

Editing Projects

This chapter describes how to edit projects in Sentaurus Workbench.

Read-Only and Writable Projects

Sentaurus Workbench projects must be placed in a disk-writable location to edit, preprocess, and execute those projects. Sentaurus Workbench recognizes a disk-writable location by analyzing user permissions of a project directory and project files.

A writable Sentaurus Workbench project must match the following criteria:

- A project directory contains the file `.project` (it is a Sentaurus Workbench project).
- A project directory is writable; users can create and delete files in the directory.
- Files in the project directory are writable. Sentaurus Workbench checks user permissions for several key project files.

A writable project in the Locked runtime editing mode can be cleaned, edited, preprocessed, and executed only if the project is not running.

A writable project in the Editable runtime editing mode can be cleaned, edited, preprocessed, and executed at any time.

Note:

Sentaurus Workbench does not require writable projects to be placed under the `STDB` file hierarchy. Attached roots can contain writable projects. Only user permissions and the runtime editing mode define the writability of a project.

Undoing Changes

To undo a change:

► Choose **Edit** > **Undo** or press Ctrl+Z.

Note:

Only tree modifications can be undone. Modified or deleted input or output files cannot be restored.

The *undo* data is not released after saving a project.

To release *undo* data, you can either:

- Choose **Project** > **Operations** > **Clean Up**, or press Ctrl+L (see Chapter 8 on page 215).
- Restart Sentaurus Workbench.

Limiting the Number of Changes

If you work with the same instance of Sentaurus Workbench for long periods, you can observe that the Sentaurus Workbench process consumes considerable memory on the host where it runs. The reason for such an increase in memory consumption is that Sentaurus Workbench retains all the changes made to the Family Tree of a project to enable Sentaurus Workbench to undo them if needed. By default, the number of changes kept in memory is unlimited.

You can reduce memory consumption by limiting the number of recent changes Sentaurus Workbench stores. To do this, set the `SWB_UNDO_STACK_SIZE` environment variable to the maximum number of changes you want Sentaurus Workbench to be able to undo. For example, the following setting restricts Sentaurus Workbench to the latest 10 changes in the Family Tree:

```
setenv SWB_UNDO_STACK_SIZE 10
```

Tools

Tools are, in general, TCAD simulation tools. Other tools can be available depending on the configuration of the tool databases of Sentaurus Workbench (see Tool Databases on page 242).

Note:

There can be multiple instances of the same tool in simulation flows.

Adding Tools to the Flow

In a new project, to add the first tool to the flow:

1. Choose **Tool** > **Add** or press the Insert key.

   The Add Tool dialog box opens (see Figure 26 on page 89).

2. On the **Tool Properties** tab, select the tool from the **Name** list or click the **Tools** button to select the tool from the Select DB Tool dialog box (see Figure 30 on page 92).

   Note:
   Sentaurus Workbench detects the tools installed and displays them on the **Available** tab of the Select DB Tool dialog box (see Figure 30 on page 92). Other tools are marked as not installed in the **Name** list and appear on the **All** tab of the Select DB Tool dialog box (see Figure 31 on page 93). You can add a non-installed tool to your simulation flow. To ensure it runs properly, install the tool appropriately and configure your `PATH` environment variable.

3. Type a unique tool label if you want to change the default label.

4. Type any command-line options if required.

5. If applicable, select a run mode from either **batch** or **interactive**.

   Sentaurus Visual has the additional **batchX** option, which allows you to export plots in the batch run.

6. Type a comment associated with the tool if required.

7. Click the **Input Files** tab (see Figure 27 on page 90).

   The table on this tab specifies all the default tool input files registered in the tool database. Each tool input file is either a user-specified file (master file) or the output file generated from the previous simulation step. Master files are usually involved in preprocessing.

8. If applicable, select whether grid and boundary files are provided as master files or produced by previous tools.

9. If applicable, select whether the tool should use common or individual parameter files.

   Each input master file has the default Sentaurus Workbench name specified in the tool database. This name cannot be changed.

10. To import the content of an existing file into the master file:

    a. Select the master file.

    b. Click **Import**, and select the file in the Open File dialog box.

Note:
The content of the imported file replaces the content of the master file. For convenience, Sentaurus Workbench creates a backup copy of the existing master file with the extension `.backup`.

11. Click the **Output Files** tab (see Figure 28 on page 90).

    The read-only table on this tab specifies the names of node files that Sentaurus Workbench generates when it preprocesses the default tool input files registered in the tool database.

12. Click the **Parallelization** tab (see Figure 29 on page 91).

    On this tab, you configure settings for parallel jobs of the tools supporting either shared-memory parallelization or message passing interface (MPI) parallelization (see Configuring the Execution of Jobs on page 189).

13. Click **OK** or **Apply**.

Figure 26 Add Tool dialog box showing Tool Properties tab

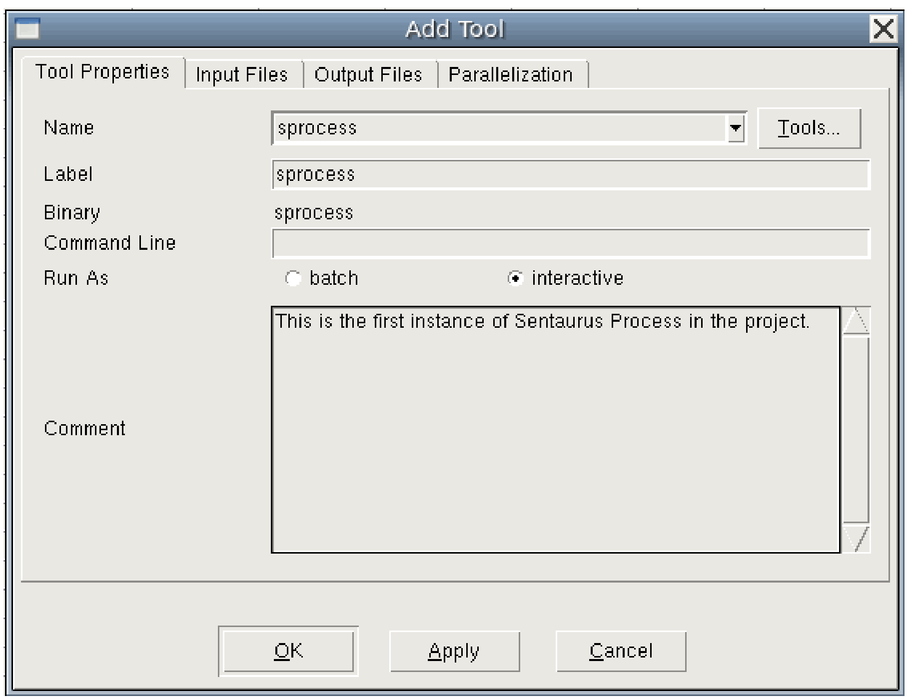


Figure 27 Add Tool dialog box showing Input Files tab

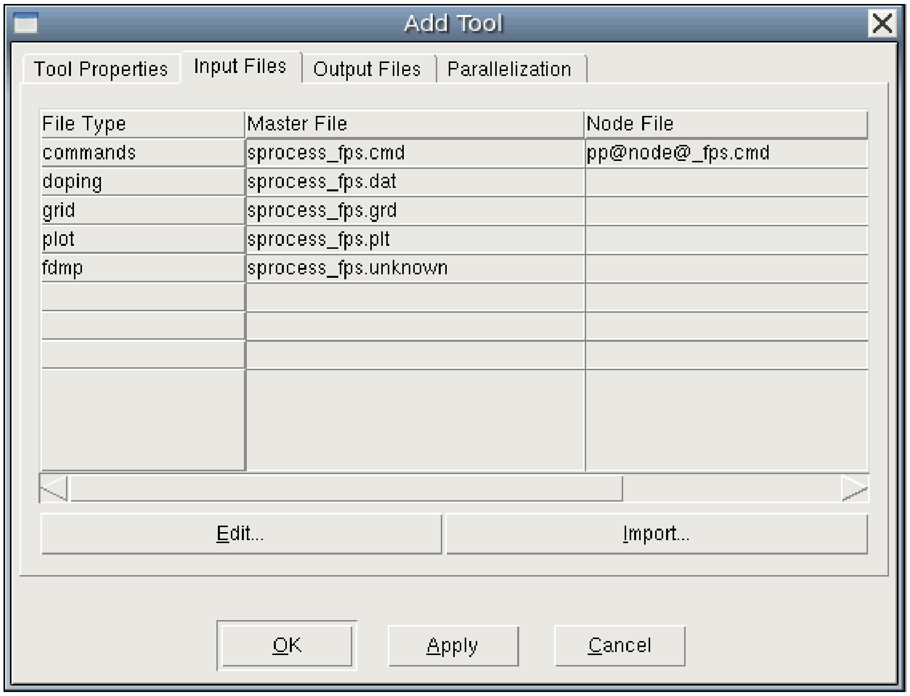


Figure 28 Add Tool dialog box showing Output Files tab

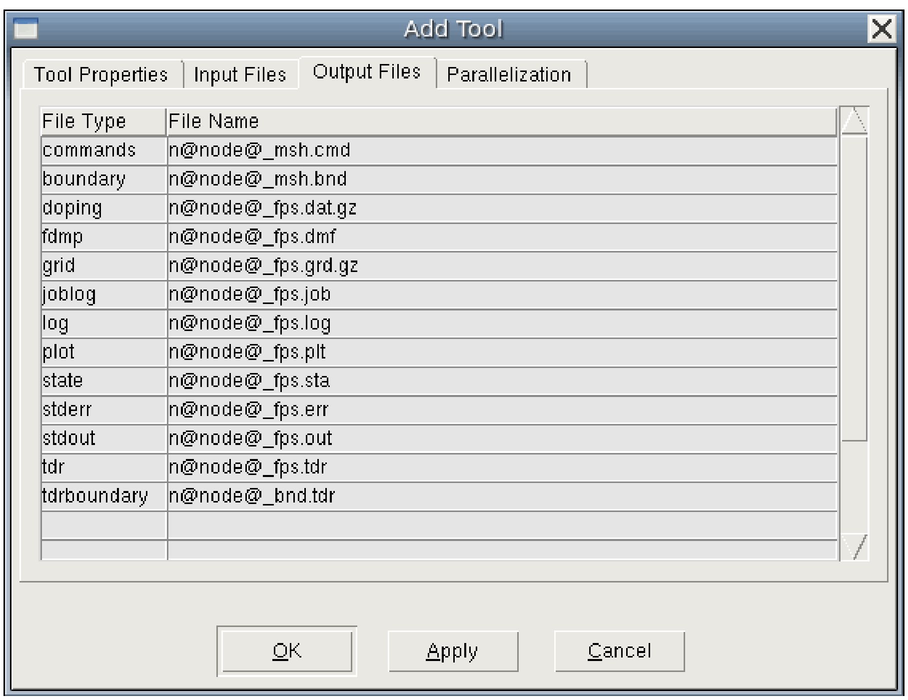


Figure 29 Add Tool dialog box showing Parallelization tab

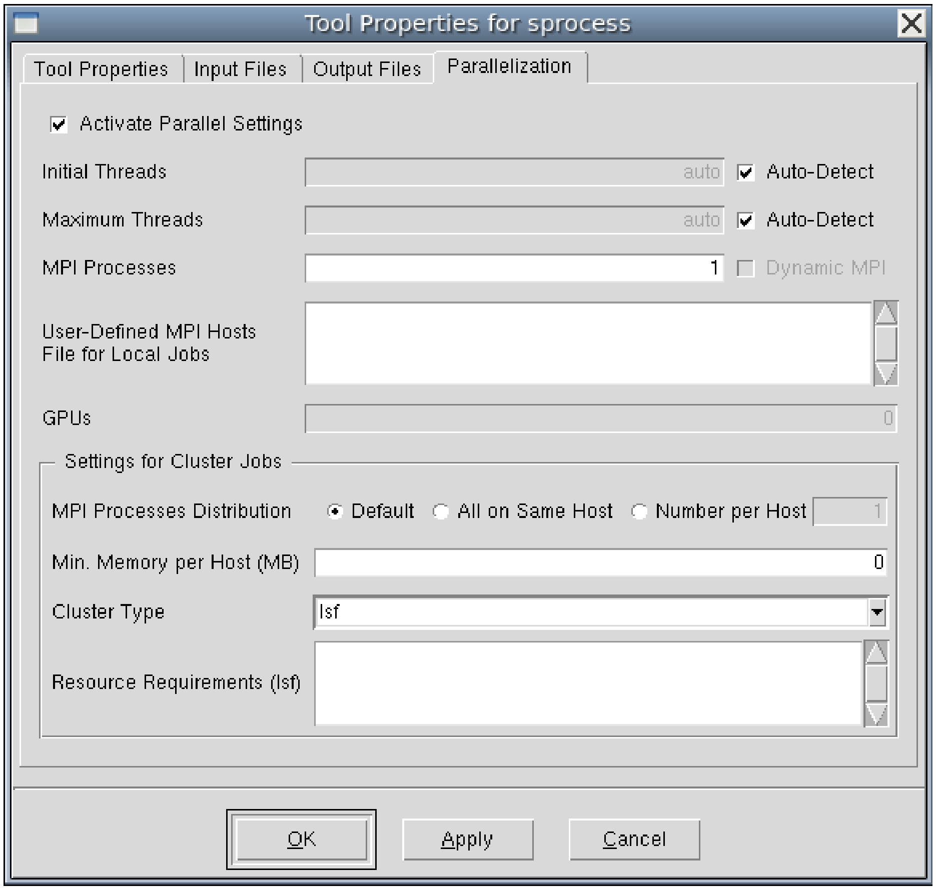


Figure 30 Select DB Tool dialog box showing available installed tools

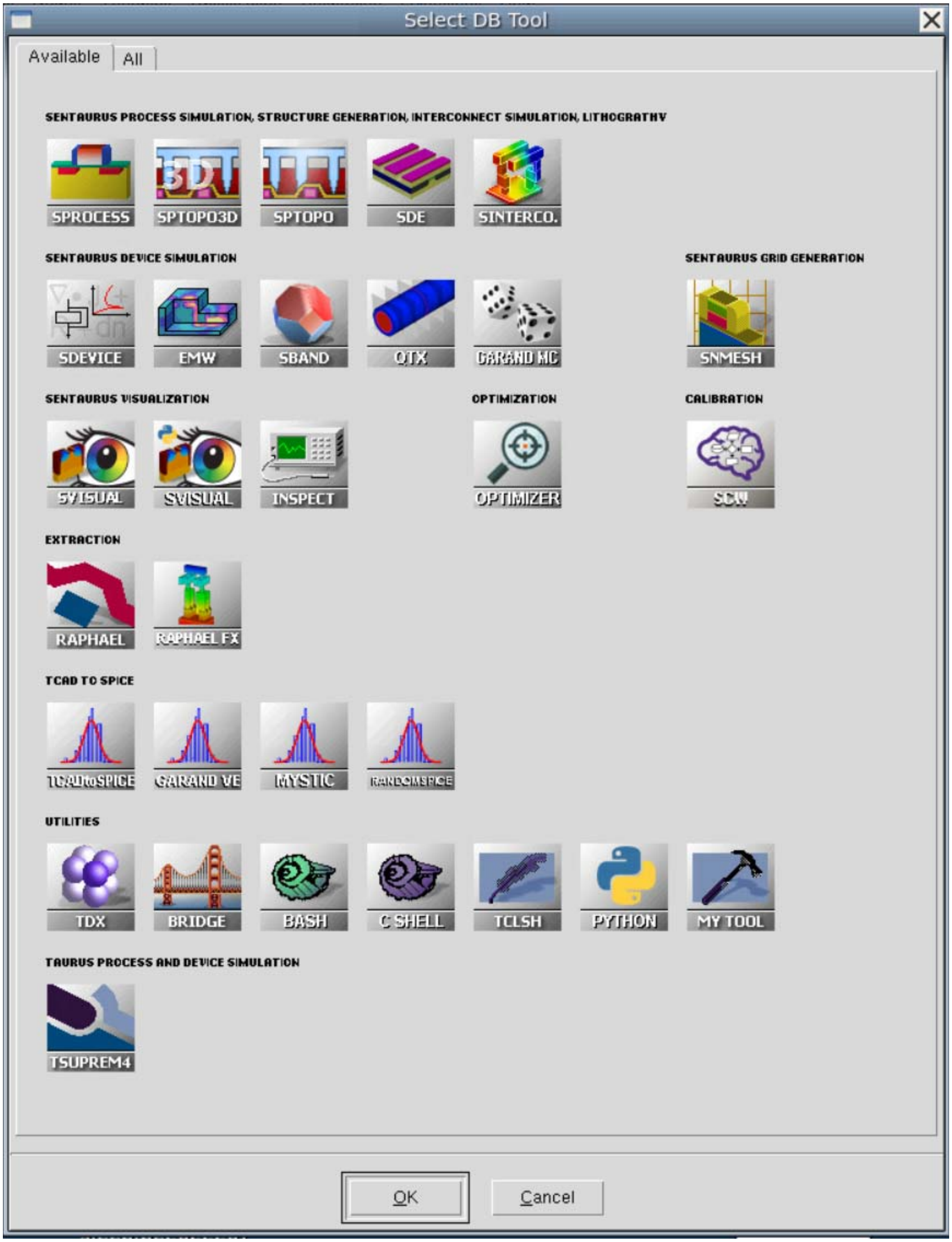
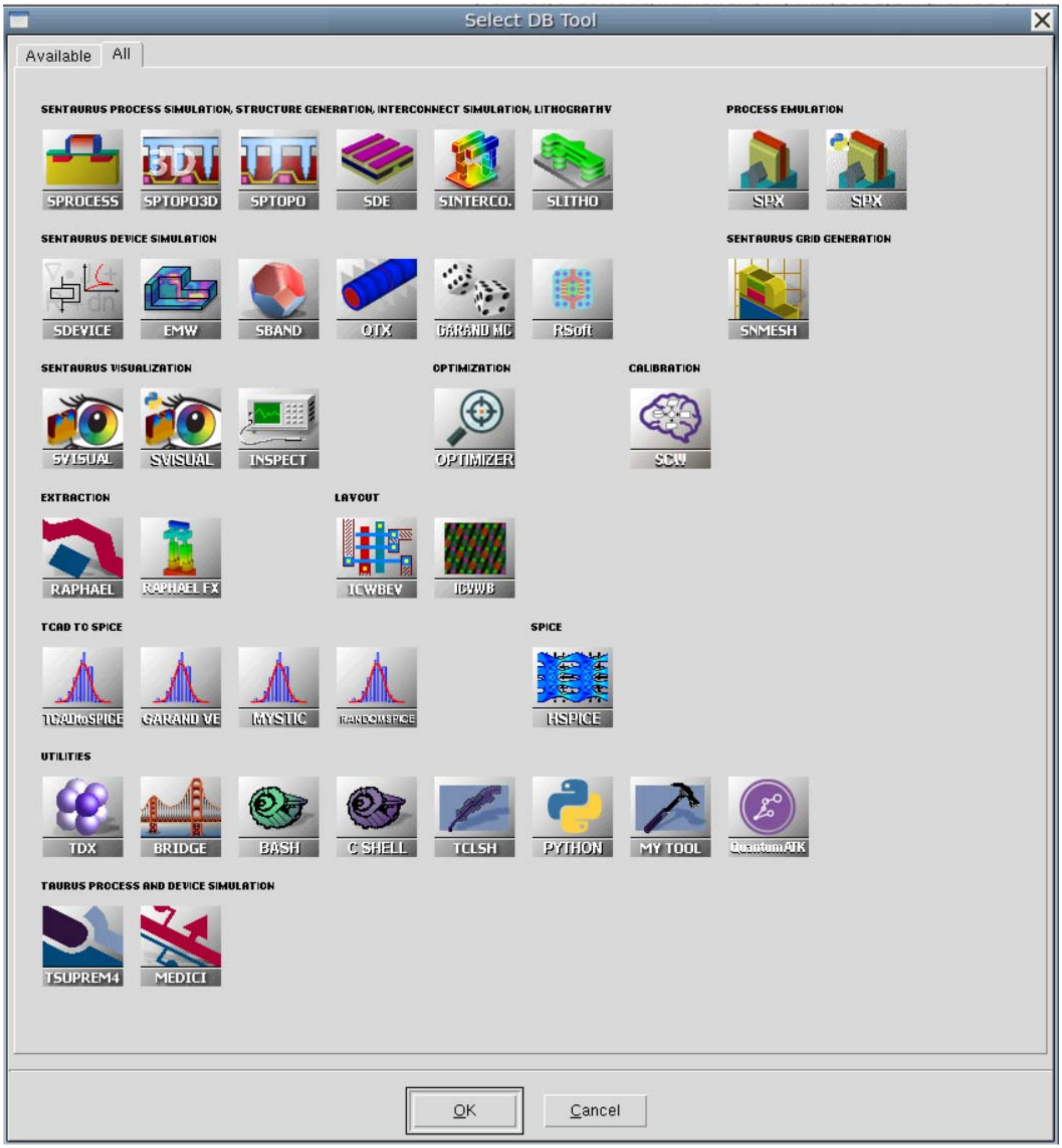


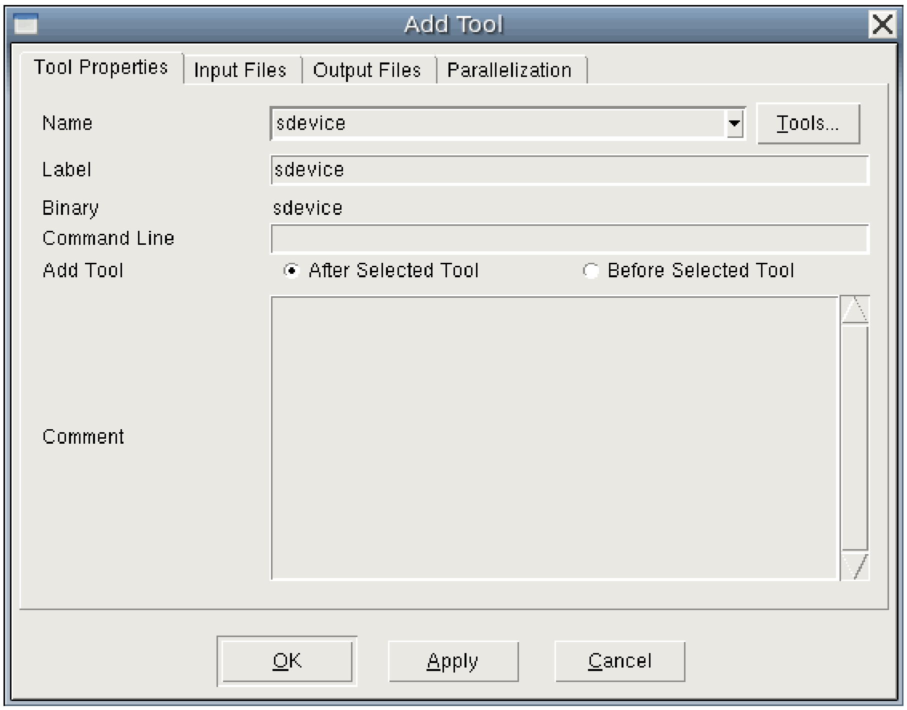
Figure 31 Select DB Tool dialog box showing all supported tools



To add subsequent tools to the flow:

1. Select a tool icon in the tool row.
2. Choose **Tool** > **Add** or press the Insert key.

   The Add Tool dialog box opens.



3. Follow Steps 2–12 for adding the first tool in the flow (see Adding Tools to the Flow on page 88).
4. On the **Tool Properties** tab, select **After Selected Tool** or **Before Selected Tool**.
5. Click **OK** or **Apply**.

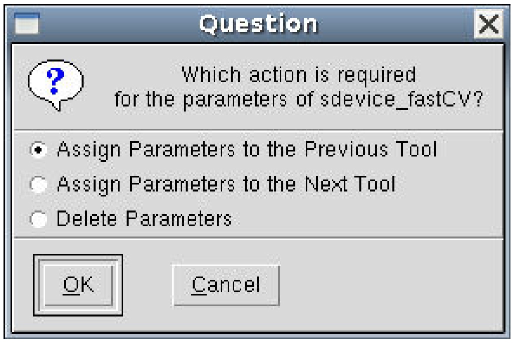
Deleting Tools From the Flow

To delete a tool instance from the flow:

1. Select the required tool icon in the tool row.

   Hold the Ctrl key during selection to select multiple tools.
2. Choose **Tool** > **Delete** or press the Delete key.

3. If the tool has parameters, specify in the dialog box whether to assign the parameters to the previous tool or the next tool, or to delete the parameters.



4. Click **OK**.

Copying Tools

You can copy tools in a project as well as between projects.

To copy a tool:

1. Select one tool or multiple tools in the flow.
2. Choose **Edit** > **Copy**, or press Ctrl+C, or right-click the tool and choose **Copy**.
3. Go to the target project (the same project or another project) and select a tool in the flow.

   New tools are inserted immediately *after* the selected tool.
4. Choose **Edit** > **Paste**, or press Ctrl+V, or right-click the tool and choose **Paste** or **Paste Special**.

New tools are inserted with their input command files and properties. Tool input files are copied from the source project to the target project.

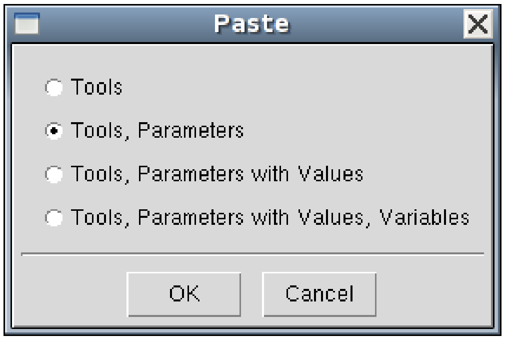
Controlling the Copying of Tools

By default, Sentaurus Workbench copies tools with their parameterization, including variables specified on the nodes of the parameterization. You can control exactly what you want to insert.

To control the copying of a tool:

1. Choose **Edit** > **Paste Special** or press Ctrl+M.

   The Paste dialog box opens.



2. Select one of the following options:

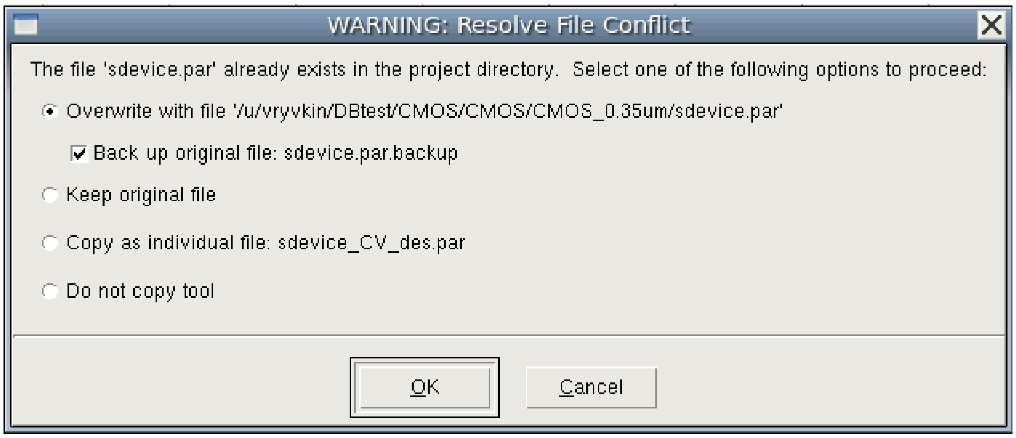
Option	What is inserted into project…
Tools	• Tool with its configuration and input files
Tools, Parameters	• Tool with its configuration and input files • Tool parameters with default values
Tools, Parameters with Values	• Tool with its configuration and input files • Tool parameters with default values • Value variation for each parameter
Tools, Parameters with Values, Variables	• Tool with its configuration and input files • Tool parameters with default values • Value variation for each parameter • User-defined variables specified on the nodes of the parameterization This is the default.

3. Click **OK**.

Resolving File Conflicts

Sentaurus Workbench copies tools and their input files including command files and parameter files. Before copying tools into the target project directory, Sentaurus Workbench checks for potential file conflicts. If the target project directory already contains a file that is included with a tool to be copied, Sentaurus Workbench displays the Resolve File Conflict dialog box (see Figure 32 on page 97).

Figure 32 Resolve File Conflict dialog box



You can select to overwrite the existing file or to keep the original file in the target project directory. If you decide to overwrite the existing file, you can instruct Sentaurus Workbench to back up this file by selecting **Back up original file**.

Some input files can be shared by several instances of the same tool in the simulation flow, for example, the Sentaurus Device parameter file. If there is a conflict with such a file, you can copy the new file as an individual file for this tool instance, which also resolves the file conflict.

Sentaurus Workbench displays the Resolve File Conflict dialog box for all conflicting files. Selecting **Do not copy tool** and clicking **OK**, or clicking **Cancel**, cancels the whole copying operation. As a result, the tool and its files are not copied into the target project directory.

Changing Tool Properties

To change tool properties:

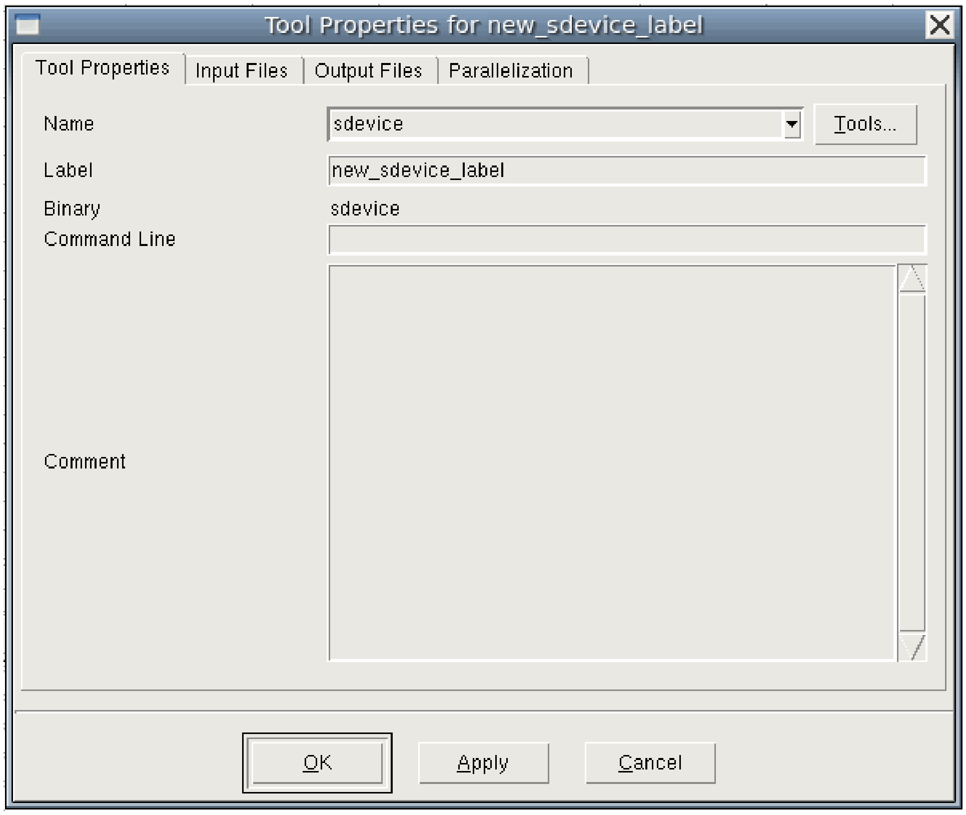
1. Select a tool in the tool row.
2. Choose **Tool** > **Properties** or double-click the tool icon.

   The Tool Properties dialog box for that tool opens (see Figure 33 on page 98).
3. Make changes as required (see Adding Tools to the Flow on page 88).
4. Click **OK** or **Apply**.

Note:

If the project is in edit mode, then you can commit changes on the **Tool Properties** tab and **Input Files** tab of the Add Tool dialog box. If the project is in running mode, then all your modifications are not supported.

Figure 33 Tool Properties dialog box for Sentaurus Device



Editing Tool Input Files

To edit tool input files:

1. Select a tool in the tool row.
2. Choose **Tool** > **Edit Input** and select the required input file.

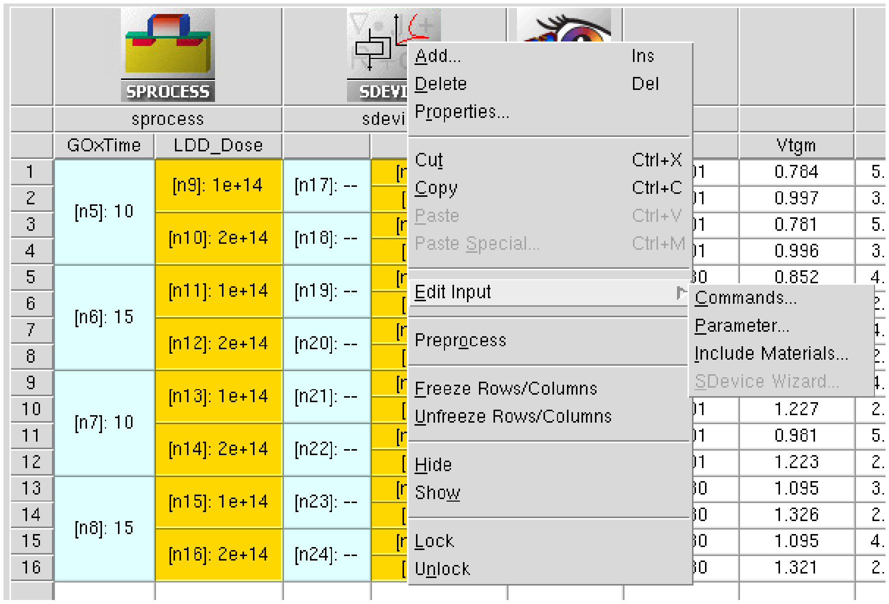
   Alternatively, right-click the tool icon and choose **Edit Input** and then the required input file (see Figure 34 on page 99).
3. Edit the file and save the changes.

Alternatively, you can use the Tool Properties dialog box to edit tool input files.

Note:

You must have write permissions for the file or project to edit files. Otherwise, the files are read only.

Figure 34 Selecting a tool input file to edit



To edit tool input files using the Tool Properties dialog box:

1. Select the required tool in the tool row.
2. Choose **Tool** > **Properties** or double-click the tool icon.

   The Tool Properties dialog box for that tool opens (see Figure 33 on page 98).
3. Click the **Input Files** tab.
4. Select the master file in the table to edit.
5. Click **Edit**.
6. Edit the file.
7. Click **OK** or **Apply**.

Editing the Parameter File for Sentaurus Device

Sentaurus Workbench allows you to edit an optional parameter file for Sentaurus Device.

To edit a Sentaurus Device parameter file:

1. Select the required Sentaurus Device tool in the tool row.
2. Choose **Tool** > **Edit Input** > **Parameter**, or right-click the tool icon and choose **Edit Input** > **Parameter**.

3. Create or edit the parameter file.
4. Click **OK**.

If the Sentaurus Device parameter file does not exist, Sentaurus Workbench displays the Create Parameter File dialog box (see Figure 35 on page 101) where you have the options to:

- Create a parameter file with the included model parameters for silicon.
- Create a parameter file with the model parameters for the selected materials.
- Create an empty parameter file and manually write it.

Note:
Selecting the **Silicon** option of the Create Parameter File dialog box is equivalent to selecting the only silicon material with the **Choose Materials** option.

The dialog box provides all the materials with their aliases specified in the Synopsys configuration database for materials (`datexcodes.txt` file). The materials are highlighted with different colors:

- Green indicates the material exists in the Sentaurus Device material database (`MaterialDB`).
- Black indicates the material does not exist in the Sentaurus Device material database.

For each selected material highlighted in green, Sentaurus Workbench creates a copy of the corresponding Sentaurus Device `MaterialDB` file in the project directory and includes this file in the generated Sentaurus Device parameter file.

For each selected material highlighted in black, Sentaurus Workbench includes a warning string in the generated Sentaurus Device parameter file. In this case, you must specify the model parameters for these materials.

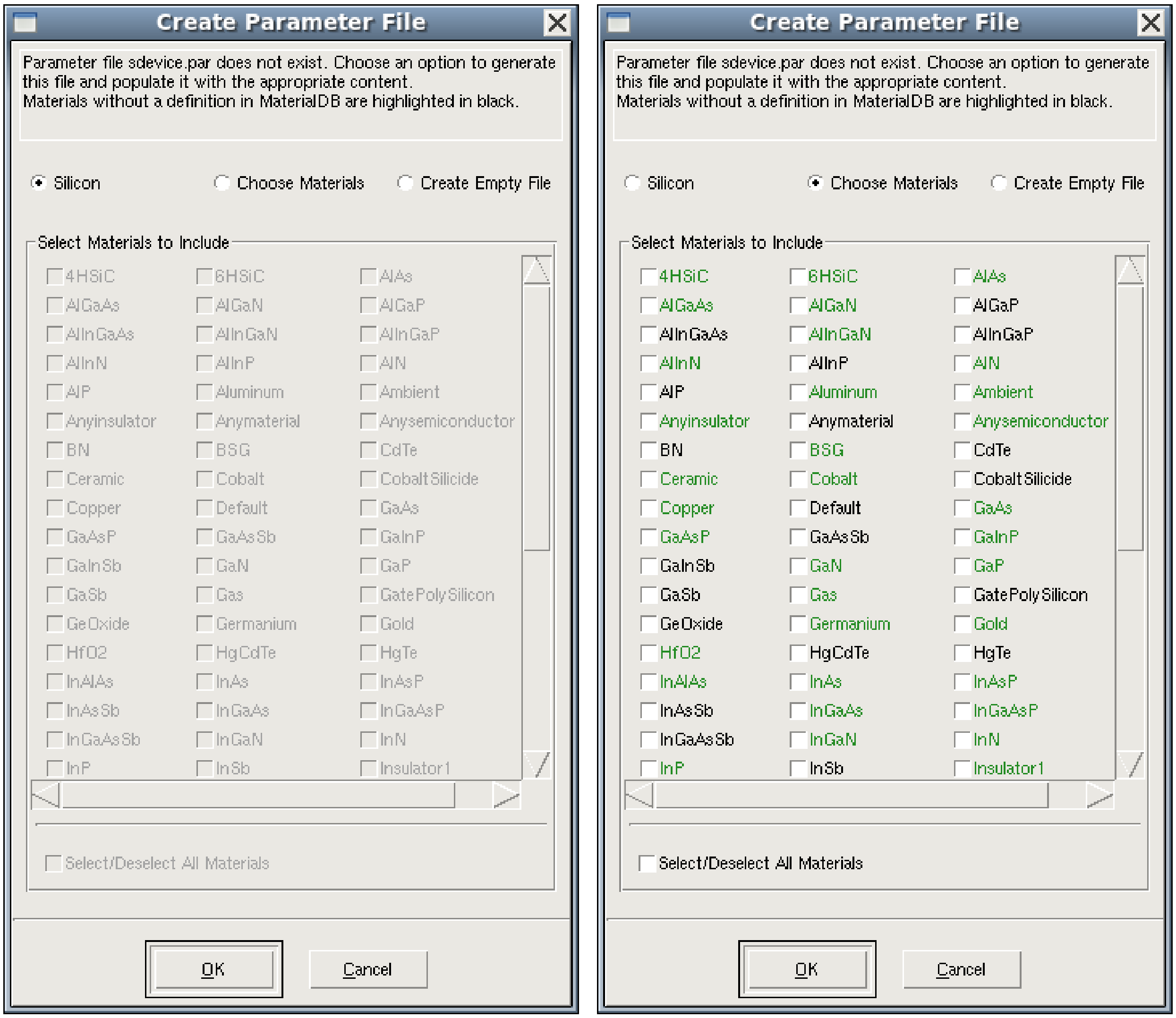
Finally, Sentaurus Workbench opens a text editor with the content of the Sentaurus Device parameter file.

For example, assume that you choose the materials GatePolySilicon, PolySi, and Silicon. The file will look like the following:

```
#define ParFileDir .
Material="GatePolySilicon" {
   # WARNING: no parameter file found for material GatePolySilicon
   # in the material database
}
Material="PolySi" {
   #includeext "ParFileDir/PolySi.par"
}
```

```
Material="Silicon" {
   #includeext "ParFileDir/Silicon.par"
}
```

Figure 35 Create Parameter File dialog box: (left) Silicon option selected and (right) Choose Materials option selected

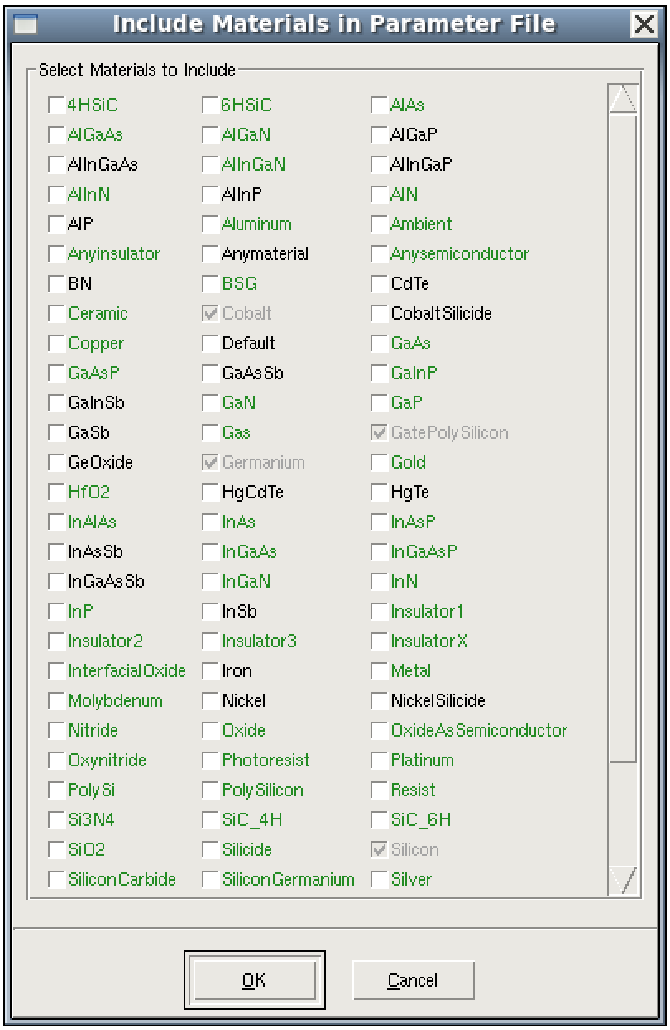


Including Additional Materials to a Parameter File

To include more materials to a Sentaurus Device parameter file:

1. Select the required Sentaurus Device tool in the tool row.
2. Choose **Tool** > **Edit Input** > **Include Materials**, or right-click the tool icon and choose **Edit Input** > **Include Materials**.

   The Include Materials in Parameter File dialog box opens.



3. Select the required materials.
4. Click **OK**.

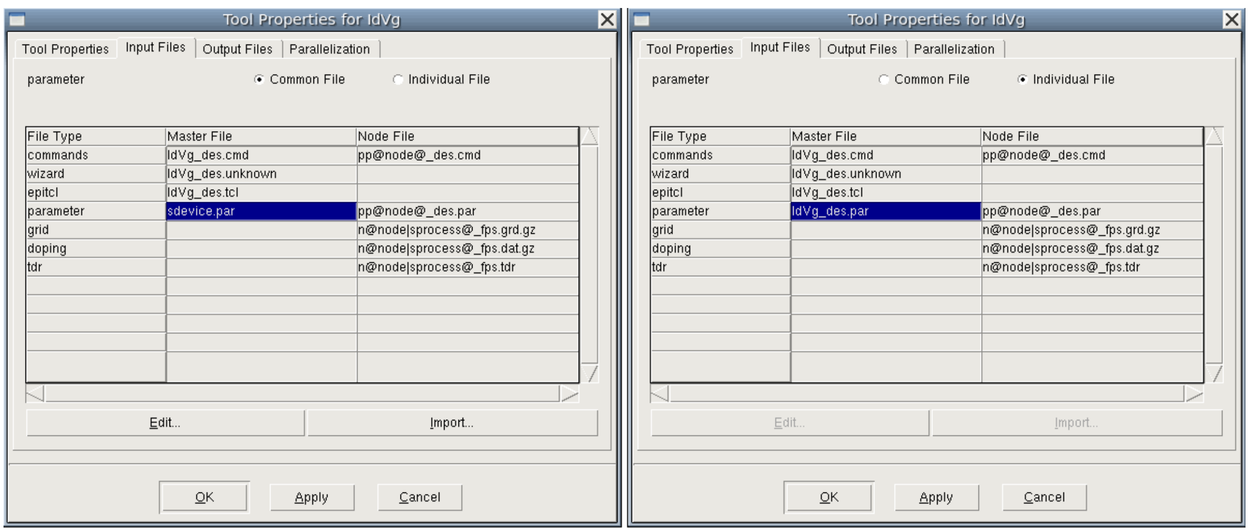
To remove materials from a Sentaurus Device parameter file:

1. Select the required Sentaurus Device tool in the tool row.
2. Choose **Tool** > **Edit Input** > **Parameter**, or right-click the tool icon and choose **Edit Input** > **Parameter**.
3. Manually remove blocks of the corresponding materials.

You can configure each Sentaurus Device tool instance in the tool row to work with either common or individual parameter files, in the Tool Properties dialog box (see Figure 36):

- Sentaurus Device tool instances with the **Common File** option share the same Sentaurus Device parameter file `sdevice.par`.
- Sentaurus Device tool instances with the **Individual File** option use their own parameter files.

Figure 36 Tool Properties dialog box for Sentaurus Device: (left) Common File option selected and (right) Individual File option selected



Included material parameter files are named differently depending on the selected option (see Table 6).

Table 6 Names of parameter file and material file

File	Name of common file	Name of individual file
Parameter file	`sdevice.par`	`<tool_label>_des.par`
Material file	`<material>.par`	`<tool_label>_<material>.par`

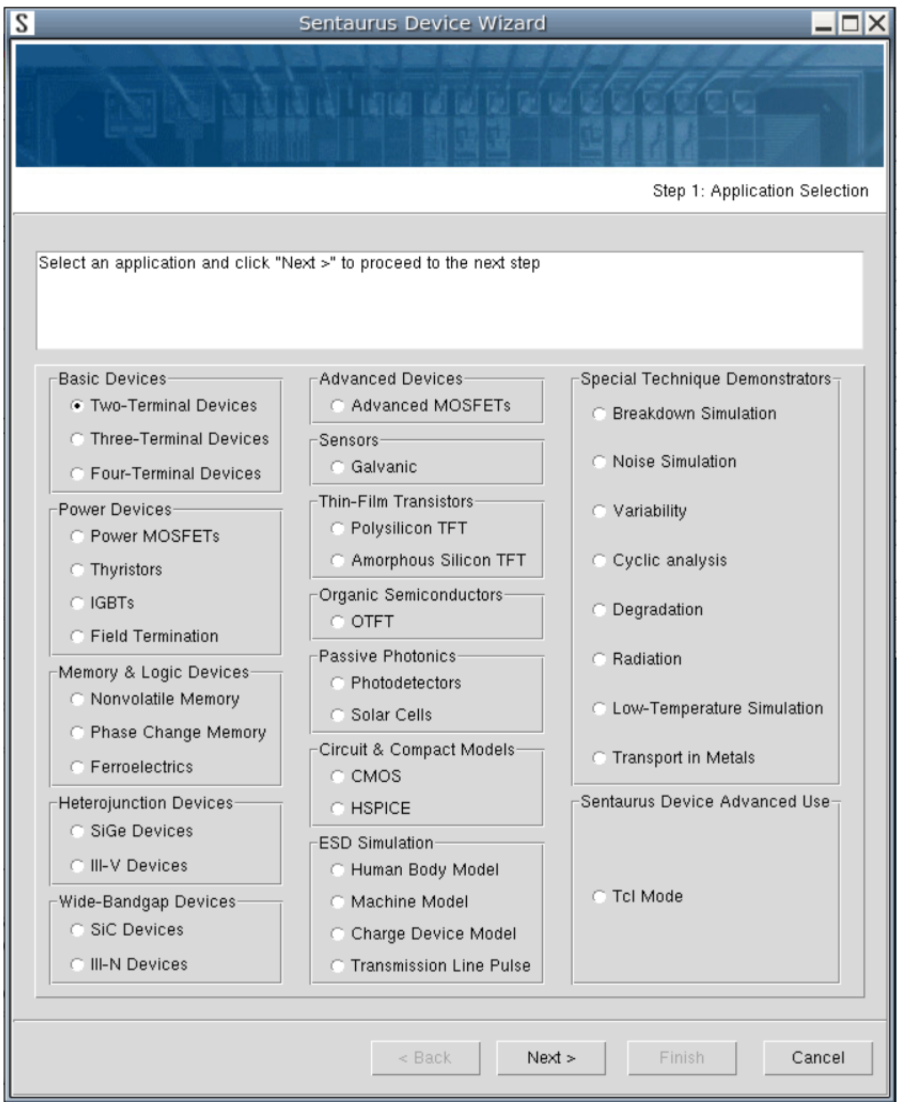
Generating Input Files for Sentaurus Device

You can generate input command files for Sentaurus Device by using the Sentaurus Device Wizard.

To launch the Sentaurus Device Wizard:

1. Select the required Sentaurus Device tool in the tool row.
2. Choose **Tool** > **SDevice Wizard**, or right-click the tool icon and choose **Edit Input** > **SDevice Wizard**.

   The user interface of the Sentaurus Device Wizard opens.



Note:

By default, the Sentaurus Device Wizard menu command is unavailable. To activate the menu command, in the SWB Preferences dialog box, set **Miscellaneous** > **Sentaurus Device Wizard** to **Yes**.

3. Proceed through the steps.
4. Click **Finish**.

For more information about the Sentaurus Device Wizard, go to the TCAD Sentaurus Tutorial, Sentaurus Workbench module, Section 3.3, Using the Sentaurus Device Wizard (see TCAD Sentaurus Tutorial: Simulation Projects on page 21).

Locking and Unlocking Tools

You can lock tool instances temporarily to prevent executing them. This means that nodes belonging to a locked tool are not involved in the next simulation run of the corresponding project. After unlocking the tool, it is a part again of the simulation project.

To lock a tool and its nodes:

1. Select the tool icon.
2. Choose **Tool** > **Lock**, or right-click the tool icon and choose **Lock**.

All nodes, real and virtual, belonging to the selected tool are locked. The icon of the locked tool changes to show a lock symbol to distinguish locked and unlocked tools (see Figure 37).

To unlock a tool and its nodes:

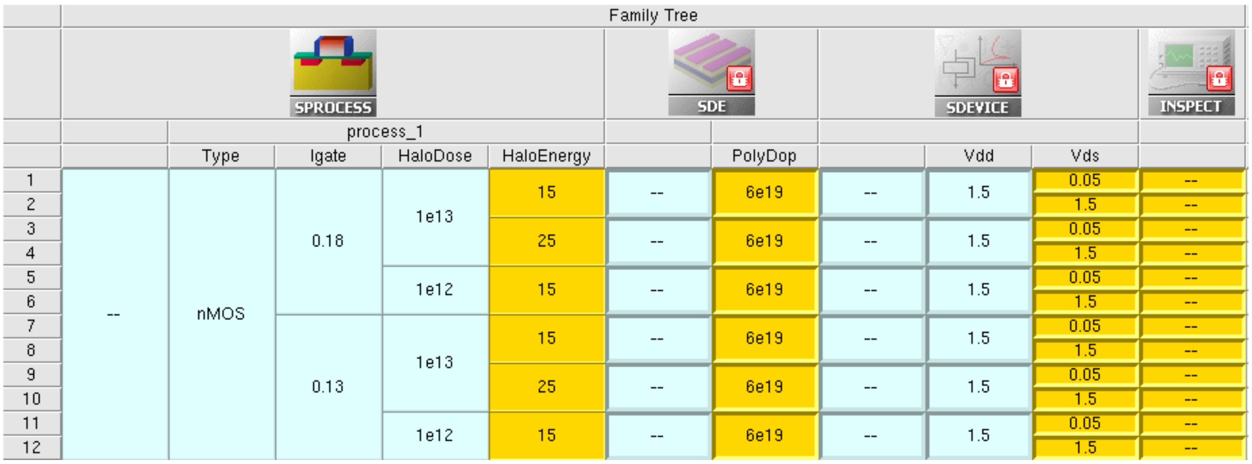
1. Select the tool icon.
2. Choose **Tool** > **Unlock**, or right-click the tool icon and choose **Unlock**.

All nodes, virtual and real, belonging to the selected tool are unlocked. The lock symbol in the tool icon disappears.

Note:

After unlocking a tool, the project should be preprocessed to guarantee that the correct node dependencies are taken into account in the next simulation run.

Figure 37 Locked and unlocked tools in the Family Tree



	Family Tree										
	SPROCESS					SDE		SDEVICE			INSPECT
		process_1									
		Type	Igate	HaloDose	HaloEnergy		PolyDop		Vdd	Vds	
1	--	nMOS	0.18	1e13	15	--	6e19	--	1.5	0.05	--
2										1.5	--
3					25	--	6e19	--	1.5	0.05	--
4										1.5	--
5				1e12	15	--	6e19	--	1.5	0.05	--
6										1.5	--
7			0.13	1e13	15	--	6e19	--	1.5	0.05	--
8										1.5	--
9					25	--	6e19	--	1.5	0.05	--
10										1.5	--
11				1e12	15	--	6e19	--	1.5	0.05	--
12										1.5	--

Configuring Double-Click Operations on Tools

You can configure the behavior of double-clicking a tool and link it to the required action.

Note:

If the double-click operation does not work on a VNC client, see Appendix D on page 332.

To configure double-click operations on tools:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** > **Double Click Action** > **On Tools**.
3. Select one of the following options:

Option	Double-clicking a tool launches
Show Properties	Tool Properties dialog box (see Changing Tool Properties on page 97)
Edit Input File	Corresponding editor for tool input command file (see Editing Tool Input Files on page 98)

4. Click **Apply**.

Parameters

A parameter is part of a tool and splits the flow at the insertion point to derive variations of that tool. Each parameter is characterized by a unique name and an arbitrary number of values.

Parameters create a family of similar simulations represented as a tree structure – the simulation tree – where levels from root to leaves (left to right) match the steps in the simulation flow.

Parameter Names

You can name parameters using the following characters:

- Alphabetic letters
- Digits
- Underscore (_)
- Hyphen (-)
- Period (.)
- Plus sign (+)

Adding Parameters to a Tool

Note:

Parameters can be added only to existing tools in a flow.

Adding the First Parameter to a Tool

To add the first parameter to a flow with only one tool:

1. Choose **Parameter** > **Add Parameter/Values** or press the Insert key.

   The Add Parameter/Values dialog box opens (see Figure 38 on page 109).
2. In the **Parameter** field, enter the name of the parameter.
3. (Optional) In the **Process Name** field, enter the name of the process.

   This field is available only for process tools.

4. Select the option for specifying values for the parameter:
   a. **List** allows you to enter values in the **List of Values** field.

      Values can be any alphanumeric character and can include + (plus), - (dash), * (start), _ (underscore), . (dot), and : (colon). Space is prohibited.

      Note:
      Do not use a double dash (--) as a parameter value. It is reserved for internal Sentaurus Workbench use.

   b. **Lin** uses linear scale to generate values. Specify the values for the various fields.
   c. **Log** uses logarithmic scale to generate values. Specify the values for the various fields.
   d. **Gaussian** uses a Gaussian function to generate values. Specify the values for the various fields.

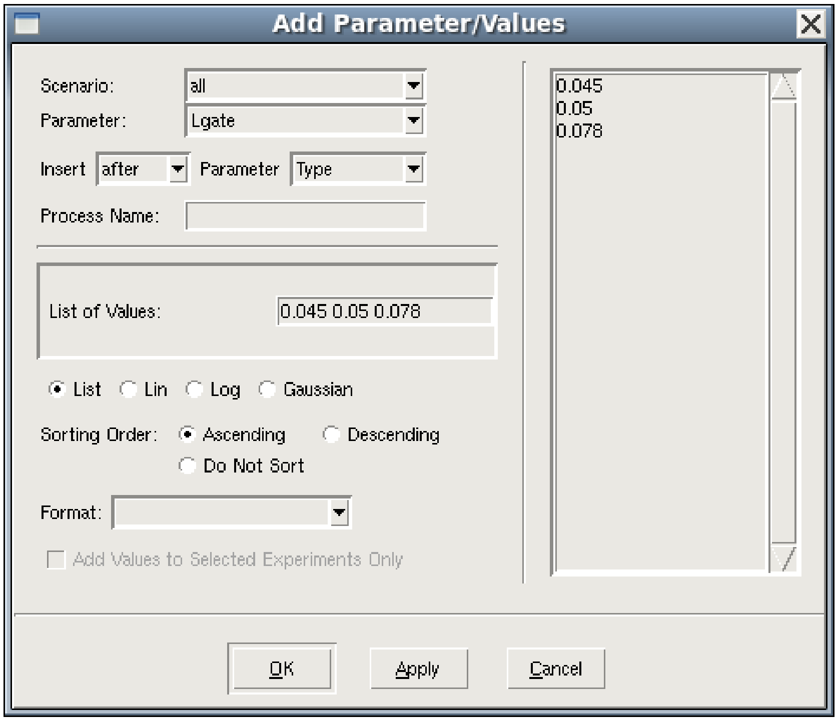
   Note:
   The first defined value becomes the *default value* of the parameter. You can change it later in the Parameter Properties dialog box.

5. Select the sorting order for the values from **Ascending**, **Descending**, or **Do Not Sort** (keeps the original order).
6. From the **Format** list, select how values will be formatted: **none** (retains the original values), **scientific**, **float**, or **integer**.
7. Check the correctness of the values in the preview pane on the right of the dialog box.

   Values are displayed as they will appear in the project flow.
8. Click **OK** to add a new parameter and to close the dialog box.

   To continue working with the Add Parameter/Values dialog box to create multiple new parameters, click **Apply**.

Figure 38 Add Parameter/Values dialog box



Adding Subsequent Parameters to a Tool

To add another parameter to a tool or to add a parameter to a tool with more than one split:

1. Select the position where the new parameter should be inserted in the parameter row.
2. Choose **Parameter** > **Add Parameter/Values** or press the Insert key.

   The Add Parameter/Values dialog box opens (see Figure 38).
3. Repeat the procedure in Adding the First Parameter to a Tool on page 107.
4. Select where to place the new parameter as follows:

   a. From the **Insert** list, select **before** or **after** the parameter chosen next.

   b. From the **Parameter** list, select the name of the parameter.

   By default, the **Insert** list shows the **after** option, that is, the default behavior is to insert a new parameter *after* the one you chose before displaying the Add Parameter/Values dialog box.
5. Click **OK** to add a new parameter and to close the dialog box.

   To continue working with the Add Parameter/Values dialog box to create multiple new parameters, click **Apply**.

Adding Values to Parameters

To add new values to an existing parameter in the flow:

1. Select the parameter in the parameter row.
2. Choose **Parameter** > **Add Parameter/Values** or press the Insert key.

   The Add Parameter Values dialog box opens.
3. Depending on the option selected, enter new values as follows:
   a. For **List**, enter the new values in the **List of Values** field.

      Values can be any alphanumeric character and can include + (plus), **-** (dash), * (start), _ (underscore), . (dot), and : (colon). Space is prohibited.

      Note:
      Do not use a double dash (--) as a parameter value. It is reserved for internal Sentaurus Workbench use.

   b. For **Lin**, enter the new values in the specified fields.
   c. For **Log**, enter the new values in the specified fields.
   d. For **Gaussian**, enter the new values in the specified fields.
4. Select the sorting order for the values from **Ascending**, **Descending**, or **Do Not Sort** (keeps the original order).
5. From the **Format** list, select how values will be formatted: **none** (retains the original values), **scientific**, **float**, or **integer**.
6. Check the correctness of the values in the preview pane on the right of the dialog box.

   Values are displayed as they will appear in the project flow.
7. Click **OK** to add the new values to the parameter and to close the dialog box.

   To continue working with the Add Parameter/Values dialog box to create new values for multiple parameters, click **Apply**.

Note:
New values are added to the existing ones for the selected parameter. Duplicates are not added. The order and the format of the new values do not affect existing values of the selected parameter.

Deleting New Values

In the Add Parameter/Values dialog box, click **Cancel** to cancel adding of new values to a parameter.

Choose **Edit** > **Undo** to revert already added values.

Limiting the Number of Values Specified for Parameters

You can unintentionally add too many values to parameters using the **Lin**, **Log**, or **Gaussian** option. For example, a typographic error using the **Lin** option can lead to a large minimum–maximum range with a small step and a huge number of values. This can take Sentaurus Workbench substantial time to accommodate so many values in the parameterization table.

You can prevent this situation by defining the maximum number of new values in the preferences. The default is 100.

To limit the number of new values for a parameter:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** > **Parameter** > **Add Parameter/Value Defaults** > **Maximum Number of Values**.
3. Specify the maximum number of new values allowed.
4. Click **Save**.

Deleting Parameters

To delete a parameter from a flow:

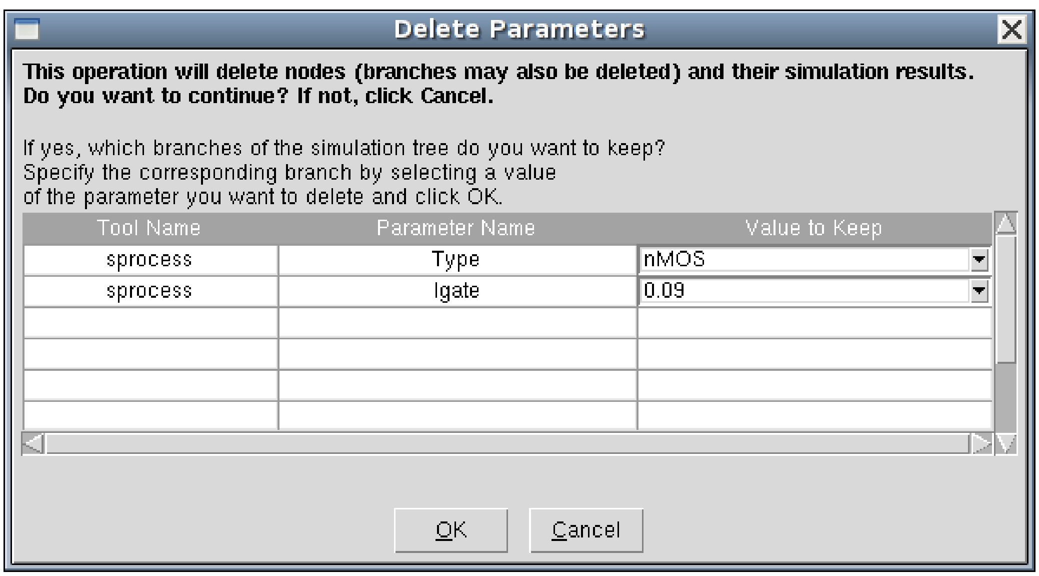
1. Select the parameter in the parameter row.
2. Choose **Parameter** > **Delete** or press the Delete key.
3. (Optional) If the parameter has more than one value, confirm which branches of the tree to keep in the Delete Parameter dialog box.

All other branches will be deleted.

4. Click **OK**.

To delete multiple parameters from a flow:

1. Select the parameters in the parameter row.
2. Choose **Parameter** > **Delete** or press the Delete key.
3. (Optional) If the parameters have more than one value, confirm the branches to retain in the Delete Parameters dialog box.



4. Click **OK**.

Copying Parameters

You can copy parameters in a project as well as between projects.

To copy parameters:

1. Select one or multiple parameters.
2. Choose **Edit** > **Copy**, or press Ctrl+C, or right-click the parameters and choose **Copy**.
3. Select the location in the target project (same project or another project) where to insert the parameters.

   When you select a tool, new parameters are added to the tool *after* existing ones. When you select a parameter, the new parameter is added immediately *after* the selected one.
4. Choose **Edit** > **Paste**, or press Ctrl+V, or right-click the parameters and choose **Paste** or **Paste Special**.

By default, parameters are copied with their parameterization, including variables specified on the nodes of the parameterization. When parameters are inserted into an empty project, Sentaurus Workbench also creates tool instances to which parameters belong.

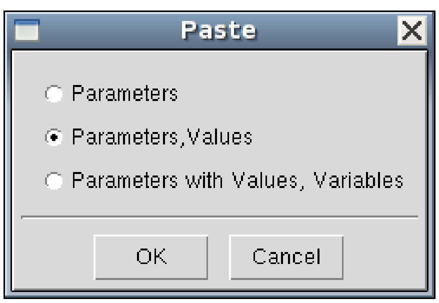
Controlling the Copying of Parameters

You can control exactly what you want to insert into a project.

To control the copying of a parameter:

1. Choose **Edit** > **Paste Special** or press Ctrl+M.

   The Paste dialog box opens.

2. Select one of the following options:

Option	What is inserted into project...
Parameters	• Parameters with default values

Option	What is inserted into project...
Parameters, Values	• Parameters with default values • Value variation for each parameter
Parameters with Values, Variables	• Parameters with default values • Value variation for each parameter • User-defined variables specified on the nodes of the parameterization This is the default.

3. Click **OK**.

Changing Parameter Properties

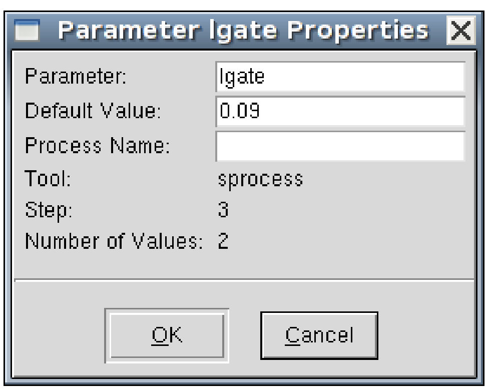
Note:

You can change the properties of parameters only if the project is in edit mode.

To change the properties of a parameter:

1. Select a parameter in the parameter row.
2. Choose **Parameter** > **Properties** or double-click the parameter.

   The Parameter Properties dialog box for that parameter opens.



3. Rename the parameter as required.

   Note:

   If you change the parameter name, it changes only in the Family Tree. All references to that parameter in input files and variables are not changed.

4. Type a new default value as required.

5. (Optional) For the parameters of process tools, change the process name as required.
6. Click **OK**.

If you want to change the default values of multiple parameters, then the most effective way is to take the values of a selected experiment (see Taking Selected Experiment as the Default on page 131).

Removing Parameter Values

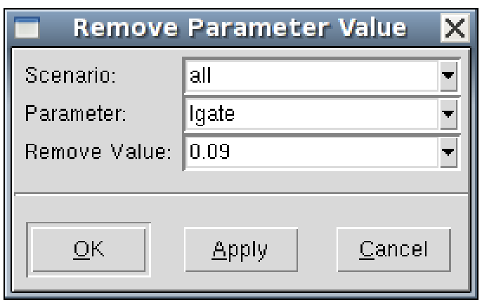
Note:

You can remove parameter values only if the project is in edit mode.

To remove a parameter value:

1. Select the parameter in the parameter row.
2. Choose **Parameter** > **Remove Value**.

   The Remove parameter Value dialog box opens.



3. Select the value to be removed.
4. Click **OK** to remove the value and close the dialog box, or click **Apply** to continue removing additional values.

Configuring Double-Click Operations on Parameters

You can configure the behavior of double-clicking a parameter and link it to the required action.

Note:

If the double-click operation does not work on a VNC client, see Appendix D on page 332.

To configure double-click operations on parameters:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** > **Double Click Action** > **On Parameters**.

3. Select one of the following options:

Option	Double-clicking a parameter launches
Show Properties	Parameter Properties dialog box (see Changing Parameter Properties on page 114)
Add Parameter Value	Add Parameter Values dialog box (see Adding Experiments on page 129)
Remove Value	Remove Parameter Value dialog box (see Removing Parameter Values on page 115)

4. Click **Apply**.

Variables

Variables are not part of the simulation. They are intended to help you interpret the results and for preprocessing.

You can define variables as either global or *per node*, where per node definitions overwrite global definitions. Per node definitions overwrite per node definitions used earlier in the tool flow (that is, closer to the root).

You can use the following variable types according to their priority (from highest to lowest):

- Extracted variables result from simulation runs. Their format in output files is `DOE: <varname> <value>` (see Extracted Variables on page 170).
- Preprocessed variables are preprocessing definitions. Their format in input files is `#set <varname> <value>` and `#seth <varname> <value>` (see Preprocessing Variables on page 169).
- Defined variables (or global variables) are defined globally for a node and its children. You can add these variables using the user interface.

Variable Names

You can name variables using the following characters, which apply to all variable types:

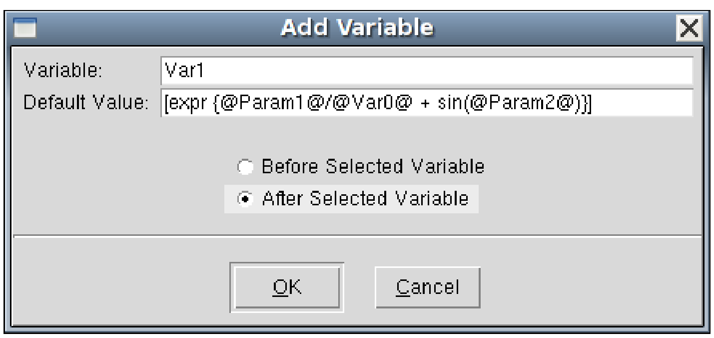
- Alphabetic letters
- Digits
- Underscore (_)
- Hyphen (-)
- Period (.)
- Plus sign (+)

Adding Global Variables

To add a new global variable to a project:

1. Choose **Variables** > **Add**, or press the Insert key.

   The Add Variable dialog box opens.

2. In the **Variable** field, enter the name of the variable.
3. Type the default value of the variable (the **Default Value** field can remain empty).

   Examples of default values are:

```
[format %f10 @param1@]

[expr 2*@param1@ + sin(@var1@)]

[if {@param1@ > 0.5} {set var1 "passed"} else {set var1 "failed"}]
```

4. (Optional) If you selected another variable in the parameter row, select the option for adding the new variable from either **Before Selected Variable** or **After Selected Variable**.
5. Click **OK**.

The default value (or formula) of a variable can be an arbitrary string, and it is used when there is no defined value per node. A Tcl parser preprocesses the value, and Tcl expressions in brackets can be used as variable values. They can have references to parameters or other variables. The references must be placed inside of a pair of @ signs.

The default value of a variable can also be used for formatting, using the Tcl `format` command, which has similar arguments as the ANSI C `printf` command.

Changing the Default Value of Global Variables

To change the default value of a global variable:

1. Select a variable in the parameter row of the Variables Values view.
2. Choose **Variables** > **Properties**, or right-click and choose **Properties**.
3. In the Variable dialog box, enter the new default value.
4. Click **OK**.

Deleting Global Variables

To delete a global variable:

1. Select a variable in the parameter row.

   Hold the Ctrl key to select multiple variables.
2. Choose **Variables** > **Delete** or press the Delete key.

Copying Global Variables

You can copy global variables in a project as well as between projects.

Note:

Only global variables can be copied.

To copy global variables:

1. Select one or more variables.
2. Choose **Edit** > **Copy**, or press Ctrl+C, or right-click the variables and choose **Copy**.
3. Select the variable in the target project (the same project or another project) where you want to insert the variables.

   New variables are added immediately *after* the selected one.
4. Choose **Edit** > **Paste**, or press Ctrl+V, or right-click the variables and choose **Paste** or **Paste Special**.

Inserted variables contain their values. A variable that is inserted into an empty project does not display its value. The variable value is visible as soon as a new tool is added to the empty project.

Formatting Variables

To format a variable:

1. Select a variable in the parameter row.
2. Choose **Variables** > **Format**.

   The Format Variable dialog box opens.
3. Select one of the predefined format options, or enter a new format using the ANSI C `sprintf` command or Tcl `format` command.

   For examples, see the rollover text available from the text box.
4. Click **OK**.

Defining Variables Per Node

If you define a variable at a particular node, it will overwrite the default value of that variable as well as any value of that variable that is defined higher in the simulation tree (that is, closer to the root).

To define a variable per node:

1. Select a node.
2. Choose **Nodes** > **Set Variable Value**.

   The Add Variable to Node dialog box opens.

3. In the **Name** field, select a variable that you want to define for this node.
4. In the **Value** field, enter a value for the variable.

   The default value of the variable is preset in the text box.
5. Click **OK**.

Note:

The Tcl parser does not parse the value of a variable at a node, unlike the default value of a variable. The value of a variable is taken as a string.

Changing and Deleting Variable Values at a Node

Note:

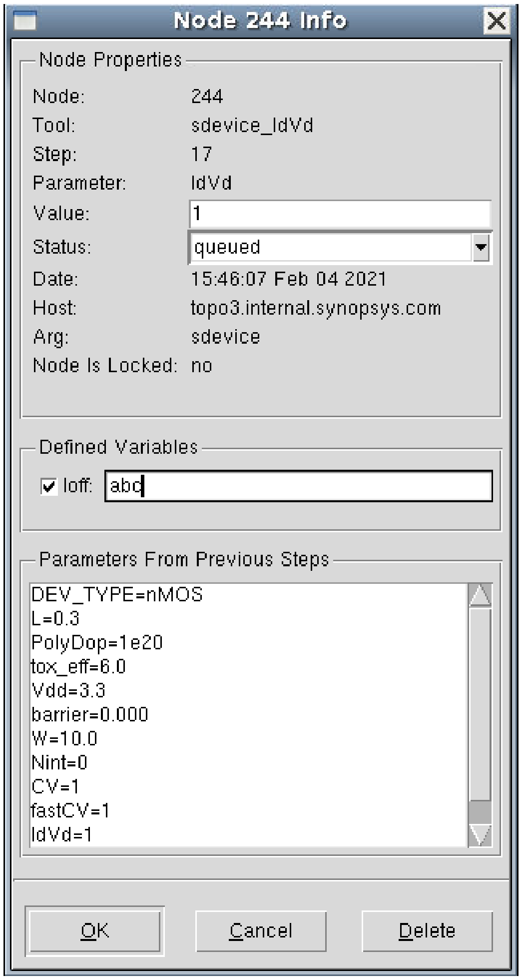
Variable values at a node can be modified only for global variables. To see which nodes have global variables, choose **View** > **Tree Options** > **Show Variables**.

To change or delete a variable value at a node:

1. Select a node.
2. Choose **Nodes** > **Edit Properties** or double-click the node.

   The Node Information dialog box opens.
3. Under Defined Variables, select the check box next to the required variable.

   Only previously defined variables are listed.



4. Enter a new value of the variable.
5. Click **OK** to change a value, or click **Delete** to delete the variable from the node.

Configuring Double-Click Operations on Variables

You can configure the behavior of double-clicking a variable and link it to the required action.

Note:

If the double-clicking operation does not work on a VNC client, see Appendix D on page 332.

To configure double-click operations on variables:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** > **Double Click Action** > **On Variables**.

3. Select one of the following options:

Option	Double-clicking a variable launches
Show Properties	Variable Properties dialog box where you can change the variable value
Specify Format	Formatting Variables dialog box (see Formatting Variables on page 119)

4. Click **Apply**.

Hiding Variables

By default, all variables are displayed in the Variables Values view, regardless of how they are defined. However, there are use cases when you might not want to see all these variables:

- You want to declare many preprocessed variables to keep your simulation setup highly parameterized. Assume you extract a few of these variables from the simulation, while the majority of variables have a single value for the entire parameterization. Displaying all the variables leads to an overcrowded Variables Values view, which makes it difficult to see the extracted variables in which you are most interested.
- You want to extract an intermediate variable in order to transfer a quantity from one tool to the next one. The actual value of this variable is not important, so you want to hide it.

To hide a variable in the Variables Values view in Sentaurus Workbench, define a preprocessed variable in an input file using the `#seth` command (see Preprocessing Variables on page 169):

```
#seth <varname> <value>
```

The only difference between the `#seth` command and the `#set` command is that `#seth` creates a variable that is not displayed in the Variables Values view. Moreover, if you define a hidden variable with the `#seth` command and then extract a value of this variable in the next node, the extracted value is not displayed in the variables table either.

The `#set` command displays the variable if it appears in a node to the right of the node where the `#seth` command appears.

Note:

You cannot hide global variables with the `#seth` command.

Nodes

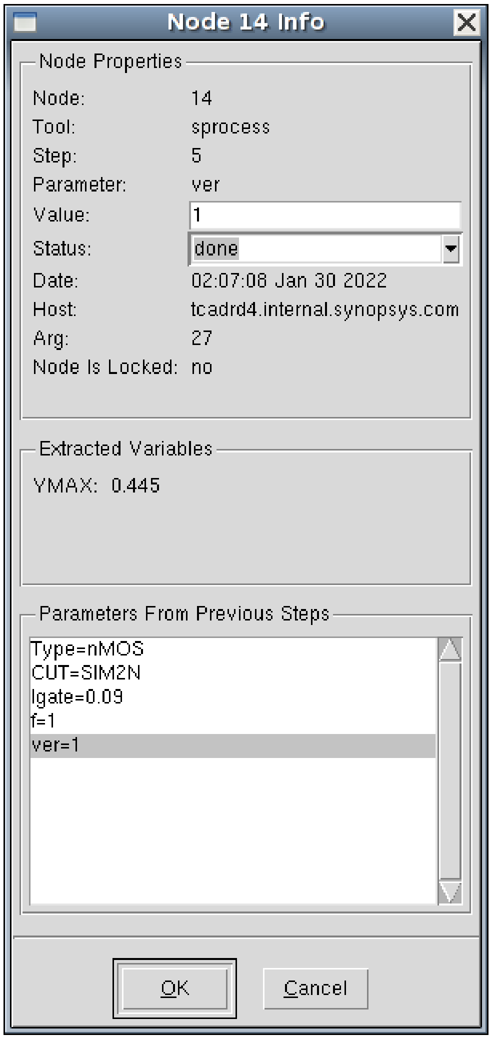
A node is a point in the Family Tree where (possibly virtual) parametric splits can occur or where a simulation tool can be changed in the tool flow. Real nodes have simulation results. In one sense, a node is an *atom* of the simulation.

Viewing and Editing Node Properties

To view and edit node properties:

1. Select a node in the flow.
2. Choose **Nodes** > **Edit Properties**.

   The Node Information dialog box opens.
3. In the **Value** field, change the value of the node.
4. From the **Status** list, change the status of the node.



5. Click **OK**.

Changing Parameter Values Directly in Node Cells

You can change parameter values directly in node cells. By default, this feature is switched off.

To switch on this feature:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** and set **Edit Value in a Cell** to **true**.
3. Click **Save**.

To edit the parameter value in a node cell:

1. Select a node in the flow.
2. Change the value in one of the following ways:
   a. Type directly in the cell.
   b. Click twice with a small time interval (this is *not* a double-click).
   c. Double-click a node.

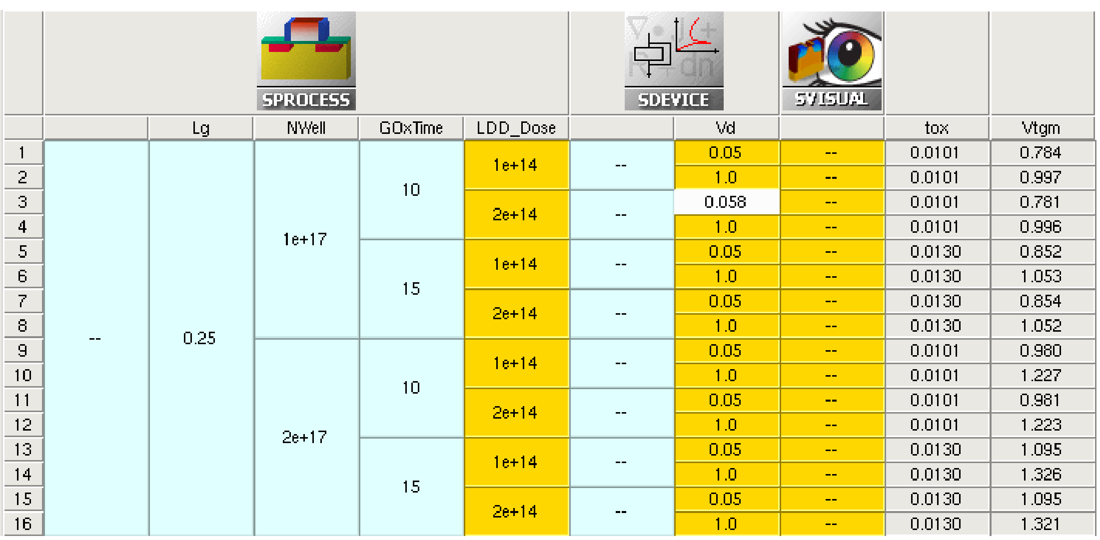
      This option is available only if the double-click operation is configured as Edit Cell (see Configuring Double-Click Operations on Nodes on page 126).
   d. Choose **Nodes** > **Edit Value**, or press the F6 key.
3. Type the necessary value.

   Use the Backspace key to delete the last symbol.
4. Press the Enter key or arrow keys, or click another node, to save the change.
5. Press the Esc key to undo the change.

Note:

Even when this feature is switched off, you can edit the parameter value of a selected node directly in the node cell by choosing **Nodes** > **Edit Value** or pressing the F6 key.

Figure 39 Editing a value directly in a node cell indicated by white cell under the Vd column



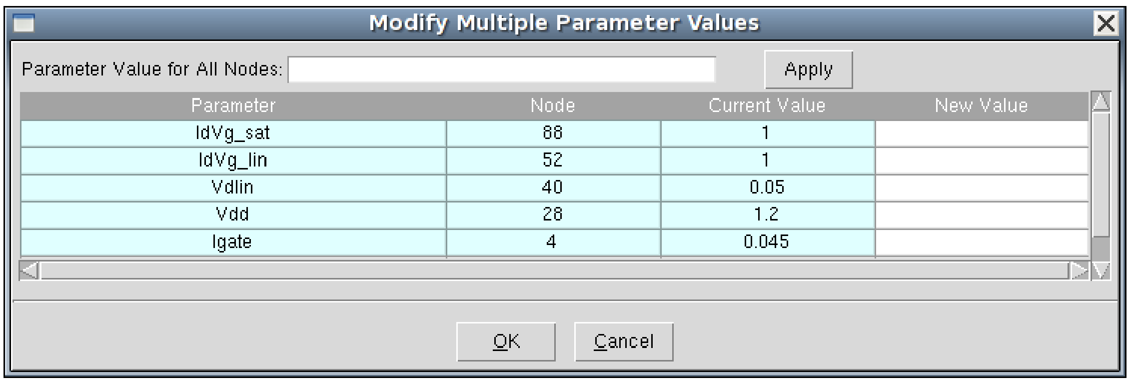
			SPROCESS				SDEVICE	SVISUAL		
		Lg	NWell	GOxTime	LDD_Dose		Vd		tox	Vtgm
1	--	0.25	1e+17	10	1e+14	--	0.05	--	0.0101	0.784
2							1.0	--	0.0101	0.997
3					2e+14	--	0.058	--	0.0101	0.781
4							1.0	--	0.0101	0.996
5				15	1e+14	--	0.05	--	0.0130	0.852
6							1.0	--	0.0130	1.053
7					2e+14	--	0.05	--	0.0130	0.854
8							1.0	--	0.0130	1.052
9			2e+17	10	1e+14	--	0.05	--	0.0101	0.980
10							1.0	--	0.0101	1.227
11					2e+14	--	0.05	--	0.0101	0.981
12							1.0	--	0.0101	1.223
13				15	1e+14	--	0.05	--	0.0130	1.095
14							1.0	--	0.0130	1.326
15					2e+14	--	0.05	--	0.0130	1.095
16							1.0	--	0.0130	1.321

Editing Parameter Values of Multiple Nodes

Sentaurus Workbench allows you to change parameter values defined on several nodes at the same time:

1. Select multiple nodes in the simulation flow.
2. Choose **Nodes** > **Modify Multiple Parameter Values**.

   The Modify Multiple Parameter Values dialog box opens.



3. Specify the necessary value for each node in the New Value column.
4. Click **OK**.

To specify the same parameter value for all nodes:

1. Type the value in the **Parameter Value for All Nodes** field.
2. Click **Apply**.
3. Click **OK**.

Configuring Double-Click Operations on Nodes

You can configure the behavior of double-clicking a node and link it to the required action.

Note:

If the double-click operation does not work on a VNC client, see Appendix D on page 332.

To configure double-click operations on nodes:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Table** > **Double Click Action** > **On Nodes**.
3. Select one of the following options:

Option	Double-clicking a node launches
Show Properties	Node Information dialog box (see Viewing and Editing Node Properties on page 123)
Edit Cell	Editor for the node cells that allows you to modify a value directly in the node cell
Launch Explorer	Node Explorer (see Node Explorer on page 76)
Run	Run Project dialog box (see Figure 64 on page 185)
Preprocess	Preprocessing of the current node.
Visualize Results	Default visualizer for viewing node output files

4. Click **Apply**.

Copying Nodes

You can copy nodes in a project and between projects.

To copy nodes:

1. Select one or multiple nodes.

   You can only make horizontal or vertical selections of nodes. Rectangular selection is not supported.
2. Choose **Edit** > **Copy**, press Ctrl+C, or right-click the nodes and choose **Copy**.
3. Select nodes in the target project (the same project or another project) where you want to insert the nodes.
4. Choose **Edit** > **Paste**, press Ctrl+V, or right-click the nodes and choose **Paste** or **Paste Special**.

Copying nodes means copying parameter values, with no change to the parameterization structure. When inserting nodes into an empty project, Sentaurus Workbench creates the necessary infrastructure: tools and parameters are copied together with the nodes.

The behavior of Sentaurus Workbench when copying nodes might differ depending on the selection of where to insert the nodes:

- Node values are replaced by copied ones when the geometry of the selected nodes is the same between the source project and the target project.
- Nodes that are copied in the tool row or the parameter row of a target project are inserted together with their parameters.

Note:

When copying nodes, the node output files are not copied to the target project.

Viewing Node Dependencies

You can view nodes that are prerequisites of selected nodes. Both implicit and explicit node dependencies are taken into account.

To view prerequisite nodes:

1. Select one or more nodes.
2. Choose **Nodes** > **Extend Selection To** > **Prerequisite Nodes**.

Sentaurus Workbench selects all nodes that are prerequisites of the originally selected ones.

Renumbering Nodes Without Cleaning Up a Project

In hierarchical projects, you can renumber nodes without cleaning up a project in one of the following ways:

- Choose **Nodes** > **Renumber All Nodes**.
- Hold the Ctrl and Alt keys, and then press the R key.
- Right-click a node and choose **Renumber All Nodes**.

Renumbering nodes is also possible when cleaning up projects.

To renumber nodes:

1. Choose **Project** > **Operations** > **Clean Up**, or press Ctrl+L.

   The Cleanup Options dialog box opens.
2. Select **Renumber the Tree**.

3. Deselect **Node Files (Output, Preprocessed)**.
4. Click **Cleanup**.

Sentaurus Workbench restores standard tree-based node numbering and ensures that node files are renamed appropriately. However, the content of node files might still contain references to previous node numbers. The next time you preprocess or run a project, the content of the node files will be updated. See Cleaning Up Projects on page 215.

Note:

For projects with traditional organization, node renumbering is impossible without losing node results.

Experiments and Scenarios

An experiment or parameter setting is a tuple that contains one value for each parameter of the flow. A scenario is a subtree of a simulation tree that contains a particular subset of experiments. Scenarios can overlap, that is, a particular node or path can be part of more than one scenario. Scenarios can be run and edited independently.

Note:

All projects have the scenario *all* that includes all the experiments of a project. Every experiment in the project is included in at least one scenario *all*. An experiment can be included in multiple scenarios.

Adding Experiments

An experiment can be added only if parameters are defined in a flow.

To add a single experiment:

1. Choose **Experiments** > **Add New Experiment**, or press the Insert key.

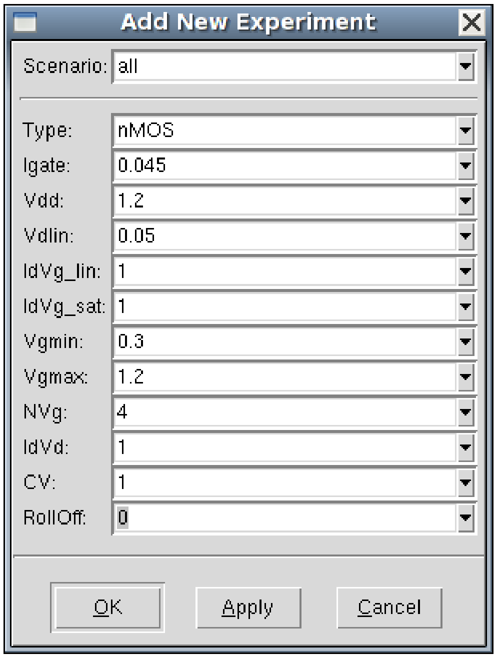
   The Add New Experiment dialog box opens.

2. From the **Scenario** list, select a scenario where the new experiment will be added.

   The default is the current scenario.

3. Select or enter values for all parameters.

   If an experiment is selected, the values are preset to those of the selected experiment.



4. Click **Apply** to add more experiments, or click **OK** to add the experiment and close the dialog box.

To add experiments by adding parameter values:

1. Choose **Experiments** > **Add Parameter/Values**.

   The Add Parameter/Values dialog box opens (see Figure 38 on page 109).

2. Select a scenario where the new experiment will be added.

   The default is the current scenario.

3. Select a parameter for which you want to add a value.

4. Select a minimum value for the parameter.

   It can be alphanumeric if only one value is added. It must be numeric if more than one value is added.

5. Repeat Steps 3–7 as specified in Adding Values to Parameters on page 110.

   If some experiments have been preselected, there is an option to add the new parameter values to all experiments (full factorial) or only to the selected experiments. By default, new values are added to selected experiments only.

It can be helpful to add a default experiment to an empty scenario. Default experiments are defined by the tuple of all project parameters with their default values. Usually, the default value of a parameter is the first value of its variation, which means the value the parameter takes in the first experiment in the Family Tree. However, you can change this in the Parameter Properties dialog box (see Changing Parameter Properties on page 114).

To create a default scenario:

1. Choose **Experiments** > **Create Default Experiment**.
2. In the Create Default Experiment dialog box, select a scenario where the new experiment will be added.
3. Click **OK**.

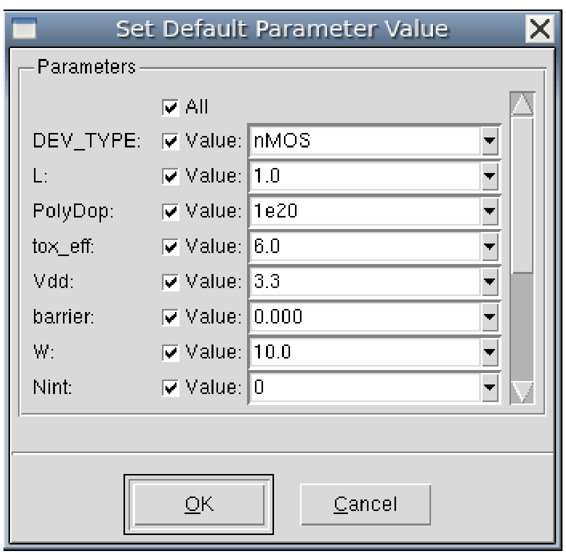
Taking Selected Experiment as the Default

You can change the default values of parameters with values defined in an experiment of your choice. It is helpful to adjust the default values of your parameters before creating designs-of-experiments with the DoE Wizard.

To change the default values of parameters:

1. Select an experiment that contains the values you want to make as the defaults.
2. Choose **Experiments** > **Take Selected Experiment As Default**.

   The Set Default Parameter Value dialog box opens.

3. Select the parameters for which you want to change default values.

   All parameters are selected by default.

4. If you want to adjust the value of a selected parameter, then enter the value explicitly.

   By default, values are preset to those of the selected experiment.

5. Click **OK** to change the default values of the selected parameters and to close the dialog box.

Excluding Experiments

Excluding experiments removes them from the current scenario but retains them in other scenarios in which they are included.

To exclude an experiment from the current scenario:

1. Select an experiment in the experiments column.
2. Choose **Experiments** > **Exclude Experiments**.

Note:

Experiments cannot be excluded from the scenario *all*. To do this, you must delete experiments.

Deleting Experiments

Deleting experiments excludes them from all project scenarios and removes them permanently from a project.

To delete experiments:

1. Select experiments in the experiments column.
2. Choose **Experiments** > **Delete Experiments**, or press the Delete key.

Note:

Apply this operation cautiously. If you want to delete experiments from the current scenario only, the correct procedure is to exclude experiments (see Excluding Experiments on page 132).

Sorting Experiments

You can sort experiments according to the values of a selected parameter.

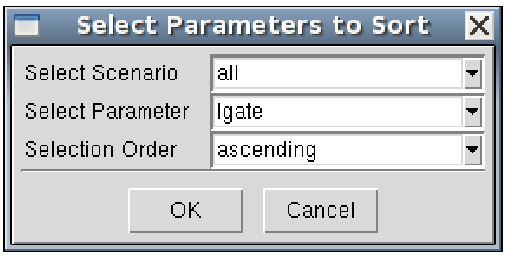
Note:

Experiments are sorted in the given order inside each subtree rather than on the entire simulation tree. That is why two experiments with the same value of the sorting parameter do not necessarily become neighbors.

To sort experiments:

1. Select experiments.
2. Choose **Experiments** > **Sort Experiments**.

   The Select Parameters to Sort dialog box opens.
3. Select the scenario, parameter, and sorting order as required.

4. Click **OK**.

Importing Experiments From a File

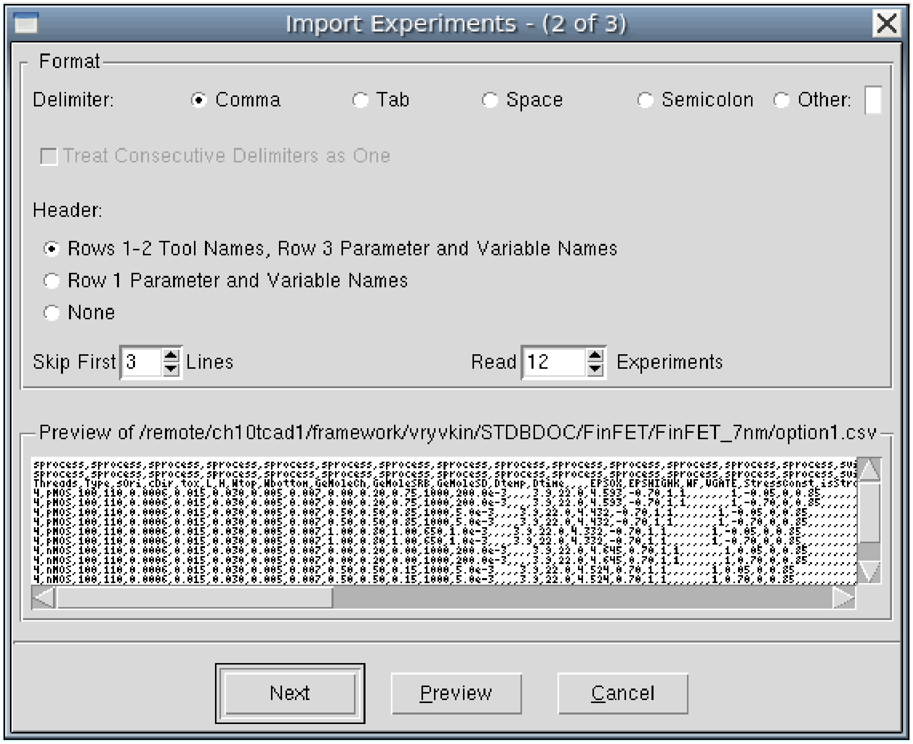
You can import experiments from a text file that contains character-separated values, for example, a `.csv` file:

To import experiments from a file:

1. Choose **Experiments** > **Import From File**.

   The Import From File dialog box opens.
2. Select the required file and click **Open**.

   The Import Experiments dialog box opens.

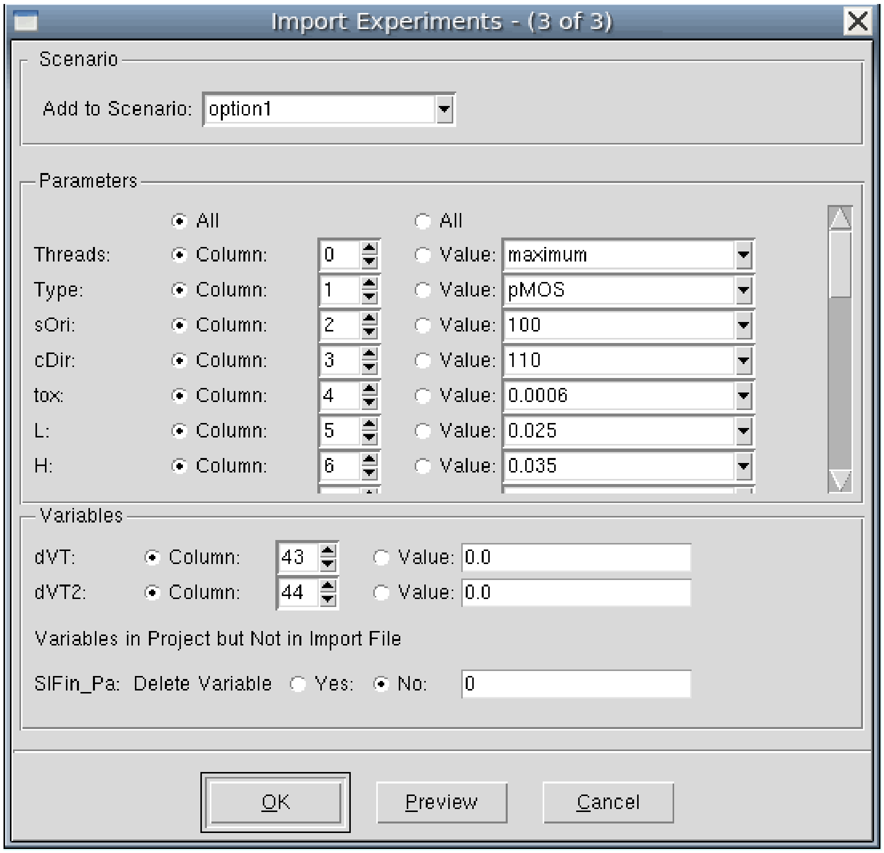


3. Select a delimiter to be used between columns in the file.
4. (Optional) Select **Treat Consecutive Delimiters as One**.

   This option is often used for space-separated values. If applicable, then Sentaurus Workbench considers multiple consecutive delimiters in the file to be one delimiter.

   The option **Treat Consecutive Delimiters as One** applies only when the **Header** is set to either **Row 1 Parameter and Variable Names** or **None**.
5. Select the instructions for the file header structure if Sentaurus Workbench does not recognize it properly (see Exporting a Spreadsheet to a Text File on page 81 for the options supported by exporting experiments).
6. From the **Skip First** list, select how many rows to omit from the beginning of the file when reading experiments.
7. From the **Read** list, select how many experiments (rows) to read from the file. This number is preset to the number of lines in the file.
8. Click **Next**.

9. Select or enter a scenario where the experiment should be imported.



10. For each parameter or variable, select either a column number in the file from which you want to take this parameter value (column numbers start from 0) or a constant value that is used for all imported experiments.

11. If a parameter or variable is in the imported file, but not in the project, then select **Add** to import the new items or **Ignore** to ignore the new items.

12. Under Parameters, select **All** to change the overall selection between a column number and a constant value.

    This option applies to both parameters and variables.

13. Click **Preview** to view which experiments will be imported.

14. Click **OK**.

Viewing Experiment Properties

To view the properties of an experiment:

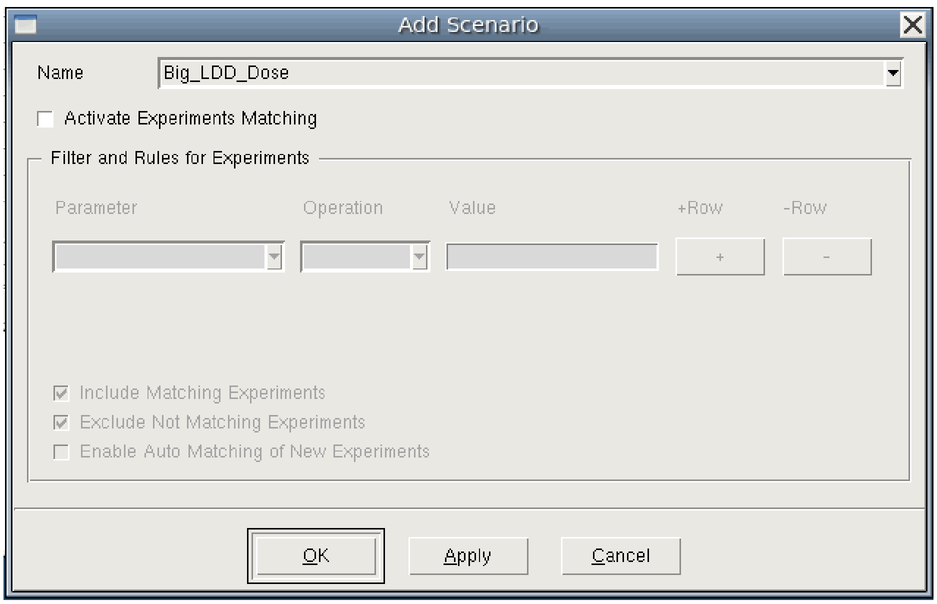
1. Select an experiment in the experiments column.
2. Choose **Experiments** > **Properties**, or double-click the experiment.

Adding Scenarios

To add a scenario to the simulation tree:

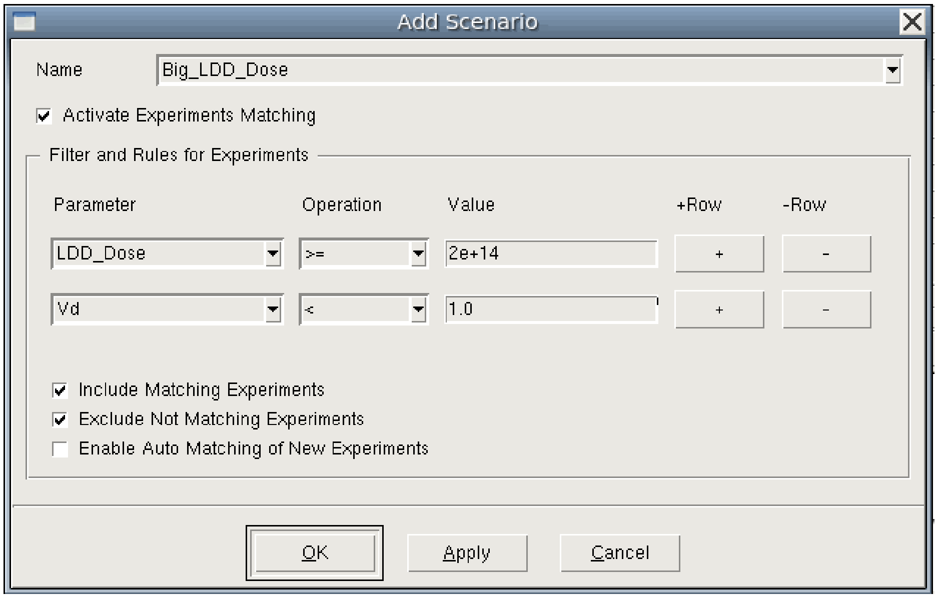
1. Choose **Scenario** > **Add**.

   The Add Scenario dialog box opens.



2. Enter a new scenario name.
3. (Optional) Select **Activate Experiments Matching** if you want to associate the scenario with specific experiments by using parameter filters.
4. From the **Parameter** box, select a parameter name.

5. From the **Operation** box, select an appropriate operation.



The following operations are supported:

- == Is equal to
- != Not equal to
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to

6. From the **Value** box, enter the appropriate value.
7. Click the **+** button to add another parameter to the filter.

   Click the **-** button to remove a parameter from the filter.

   Note:
   Sentaurus Workbench combines the given parameter comparisons by using the logical *AND* operation. For example:

   ```
   LDD_Dose >= 2e+14 && Vd < 1.0
   ```

8. Select **Include Matching Experiments** if you want the new scenario to include all experiments that match the filter specified.
9. Select **Exclude Not Matching Experiments** if you want the new scenario to exclude all experiments that do not match the filter specified.

10. Select **Enable Auto Matching of New Experiments** if you want Sentaurus Workbench to retain control over the new scenario.

    When a new experiment is added, Sentaurus Workbench automatically includes it in this scenario if the experiment passes the scenario filter. If an experiment no longer matches the scenario filter after you changed its parameter value, Sentaurus Workbench excludes this experiment from the scenario.

11. Click **OK** or **Apply**.

Changing Scenario Properties

To change scenario properties:

1. Switch to a scenario in the Sentaurus Workbench table.
2. Choose **Scenario** > **Properties**.

   The Scenario Properties dialog box for this scenario opens.

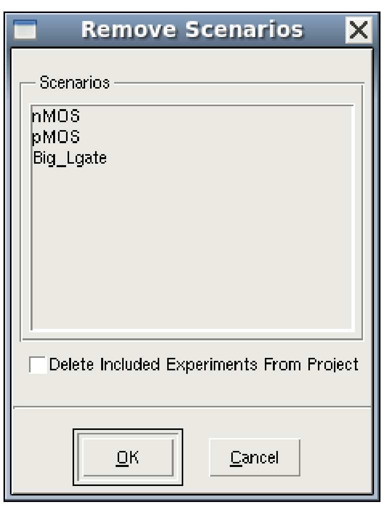
3. Make changes as required (see Adding Scenarios on page 136).
4. Click **OK** or **Apply**.

Removing Scenarios

To remove a scenario from the simulation tree:

1. Choose **Scenario** > **Remove**.

   The Remove Scenarios dialog box opens.



2. Select a scenario.

   Hold the Ctrl key during selection to select multiple scenarios.

3. (Optional) Select **Delete Included Experiments From Project**.

   If you select this option, the included experiments are deleted permanently from the project.

   By default, experiments are excluded only from the scenarios you delete and are included in the scenario *all*.

4. Click **OK**.

Including Experiments in Different Scenarios

You can manage which experiments are included in different scenarios.

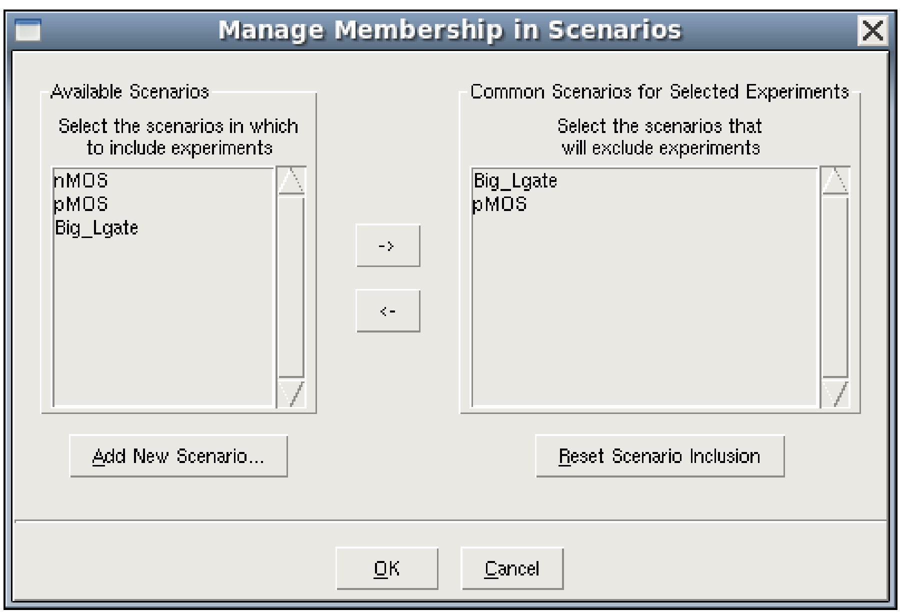
To include experiments into different scenarios:

1. Select the experiments in the experiments column.

   Hold the Ctrl key during selection to select multiple experiments.

2. Choose **Experiments** > **Manage Membership in Scenarios**.

   The Manage Membership in Scenarios dialog box opens.



3. In the **Common Scenarios for Selected Experiments** field, list the scenarios in which you want to include selected experiments.

   Any scenario added to this field will include all the selected experiments.

4. Add scenarios from the **Available Scenarios** field to the **Common Scenarios** field by clicking the **->** button.

   Note:
   The **Available Scenarios** field does not show dynamic scenarios, that is, scenarios you created with the **Enable Auto Matching of New Experiments** option. Sentaurus Workbench controls the content of these scenarios automatically, and you cannot manually include or exclude experiments to or from them.

5. Remove scenarios from the **Common Scenarios** field by clicking the **<-** button.

   If a scenario does not exist yet, click **Add New Scenario** to create it, and then add it to the **Common Scenarios** field.

6. Click **OK**.

When you click **OK**, Sentaurus Workbench ensures that selected experiments are included in all the scenarios listed in the **Common Scenarios** field. Sentaurus Workbench excludes the selected experiments from all other scenarios that are not listed in the **Common Scenarios** field.

Another option is to copy or move experiments between scenarios.

To copy experiments between scenarios:

1. Select the experiments to be copied in the experiments column.

   Hold the Ctrl key during selection to select multiple experiments.

2. Choose **Edit** > **Copy** to copy experiments, or choose **Edit** > **Cut** to move experiments.
3. Select the scenario where you want to copy experiments from the scenario box in the toolbar, or choose **Scenario** > **Next** or **Scenario** > **Previous**.
4. Choose **Edit** > **Paste**.

Excluding Experiments From Scenarios

To exclude experiments from dedicated scenarios:

1. Select the experiments to be included in the experiments column.

   To select multiple experiments, hold the Ctrl key during selection.

2. Choose **Experiments** > **Manage Membership in Scenarios**.

   The Manage Membership in Scenarios dialog box opens (see Including Experiments in Different Scenarios on page 139).

3. In the **Common Scenarios** field, remove the scenarios that should not have the selected experiments.
4. (Optional) Click **Reset Scenario Inclusion** to exclude the selected experiments from all the common scenarios and to clear the **Common Scenarios** field.
5. (Optional) Click **Add New Scenario** to create a new empty scenario in the **Available Scenarios** field.
6. Click **OK**.

When you click **OK**, Sentaurus Workbench ensures that selected experiments are included in all the scenarios listed in the **Common Scenarios** field. Sentaurus Workbench excludes the selected experiments from all other scenarios that are not listed in the **Common Scenarios** field.

Copying and Moving Experiments Between Projects

To copy and move experiments between projects if both projects have same number of parameters in the flow:

1. Select the experiments to be copied in the experiments column.

   Hold the Ctrl key during selection to select multiple experiments.

2. Choose **Edit** > **Copy** to copy experiments, or choose **Edit** > **Cut** to move experiments.
3. Open a project where you want to copy experiments in the same instance of Sentaurus Workbench.
4. Select the scenario where you want to copy experiments from the scenario box in the toolbar, or choose **Scenario** > **Next**, or **Scenario** > **Previous**.
5. Choose **Edit** > **Paste**.

Note:

When copying experiments into an empty project, Sentaurus Workbench creates the corresponding infrastructure (tool steps and parameters).

When copying experiments into a project with experiments, the experiments are copied only if the source and target projects have the same number of parameters and the parameters in both projects have the same names and appear in the same order.

You cannot manually copy and move experiments to and from dynamic scenarios, that is, scenarios you created with the **Enable Auto Matching of New Experiments** option. Sentaurus Workbench controls the content of these scenarios automatically.

Pruning and Unpruning

Pruning removes superfluous paths from a simulation tree. An entire subtree can be removed, reducing the simulation tree.

Note:

Pruning and unpruning are not as important as in previous versions of Sentaurus Workbench, since it is no longer necessary to use a full factorial tree. This feature is provided for backward compatibility and special applications that require partial trees.

To prune a simulation tree:

1. Select the nodes.

   The entire subtree starting with these nodes will be pruned.

2. Choose **Nodes** > **Configuration** > **Prune**, or press Ctrl+E.

To unprune a tree:

1. Select the nodes.

   The entire subtree starting with these nodes will be unpruned.

2. Choose **Nodes** > **Configuration** > **Unprune**, or press Ctrl+U.

To show or hide pruned nodes:

► Choose **View** > **Tree Options** > **Show Pruned**, or press the F8 key.

Locking Nodes

Node locking is useful for large projects where preprocessing all nodes is cumbersome. In addition, if you need to keep the current status of the preprocessing results for a specific tool, scenario, or node, the corresponding nodes can be locked. Locking prevents nodes from being preprocessed. These nodes can be unlocked if preprocessing is required.

To lock nodes:

1. Select the nodes.
2. Choose **Nodes** > **Configuration** > **Lock**.

To unlock nodes:

1. Select the nodes.
2. Choose **Nodes** > **Configuration** > **Unlock**.

Quick-Running Nodes

The quick-running of nodes submits nodes directly to the queue specified, and nodes are run based on the function of the queue. The project is set to a running state.

To quick-run a node:

1. Select the nodes.
2. Choose **Nodes** > **Quick Run**, and select the required queue.

Note:

The queue list is loaded from the global queue definition file or the site queue definition file, if it is specified by the `SWB_SITE_SETTINGS_DIR` environment variable.

Folding and Unfolding Nodes

It can be difficult to navigate projects having large tables with thousands of experiments. You can reduce the size of the table by folding multiple experiments into one experiment. You can apply folding to split points of the parameterization tree. Where new experiments appear, these are nodes having two or more child nodes.

To fold nodes:

1. Select the nodes.

   The entire subtree starting with these nodes will be folded.
2. Choose **Nodes** > **Configuration** > **Fold**.

Figure 40 shows how the view of the project tree changes after two folding operations:

- Folding nodes 6 and 44
- Folding nodes 2 and 3

The fold operation is applied from left to right. If the selected node does not have direct child nodes, Sentaurus Workbench tries to fold all the nodes to the right of this node.

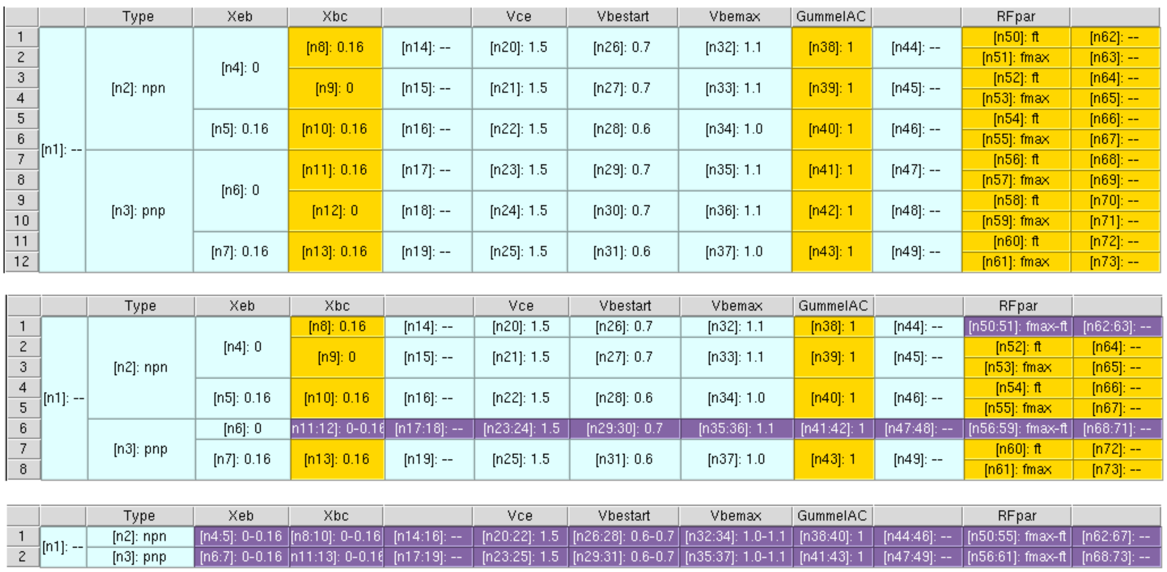
Nodes belonging to collapsed experiments represent several nodes and are shown with the specific *folded* status. These nodes display a range of values of all the nodes behind them.

You can apply almost all operations to folded nodes, that is, you can preprocess, run, terminate, and clean up nodes, and you can visualize node results. You can consider the folded node as a multiple-node selection that includes the folded node and all the nodes hidden behind it.

Note:

Folding nodes does not remove any experiments or result files. The fold operation simply hides nodes behind the folded node. You can display hidden nodes and experiments at any time using the unfold operation.

Figure 40 Original and folded project tree



		Type	Xeb	Xbc		Vce	Vbestart	Vbemax	GummelAC		RFpar	
1	[n1]: --	[n2]: npn	[n4]: 0	[n8]: 0.16	[n14]: --	[n20]: 1.5	[n26]: 0.7	[n32]: 1.1	[n38]: 1	[n44]: --	[n50]: ft	[n62]: --
2											[n51]: fmax	[n63]: --
3				[n9]: 0	[n15]: --	[n21]: 1.5	[n27]: 0.7	[n33]: 1.1	[n39]: 1	[n45]: --	[n52]: ft	[n64]: --
4											[n53]: fmax	[n65]: --
5			[n5]: 0.16	[n10]: 0.16	[n16]: --	[n22]: 1.5	[n28]: 0.6	[n34]: 1.0	[n40]: 1	[n46]: --	[n54]: ft	[n66]: --
6											[n55]: fmax	[n67]: --
7		[n3]: pnp	[n6]: 0	[n11]: 0.16	[n17]: --	[n23]: 1.5	[n29]: 0.7	[n35]: 1.1	[n41]: 1	[n47]: --	[n56]: ft	[n68]: --
8											[n57]: fmax	[n69]: --
9				[n12]: 0	[n18]: --	[n24]: 1.5	[n30]: 0.7	[n36]: 1.1	[n42]: 1	[n48]: --	[n58]: ft	[n70]: --
10											[n59]: fmax	[n71]: --
11			[n7]: 0.16	[n13]: 0.16	[n19]: --	[n25]: 1.5	[n31]: 0.6	[n37]: 1.0	[n43]: 1	[n49]: --	[n60]: ft	[n72]: --
12											[n61]: fmax	[n73]: --

		Type	Xeb	Xbc		Vce	Vbestart	Vbemax	GummelAC		RFpar	
1	[n1]: --	[n2]: npn	[n4]: 0	[n8]: 0.16	[n14]: --	[n20]: 1.5	[n26]: 0.7	[n32]: 1.1	[n38]: 1	[n44]: --	[n50:51]: fmax-ft	[n62:63]: --
2				[n9]: 0	[n15]: --	[n21]: 1.5	[n27]: 0.7	[n33]: 1.1	[n39]: 1	[n45]: --	[n52]: ft	[n64]: --
3											[n53]: fmax	[n65]: --
4			[n5]: 0.16	[n10]: 0.16	[n16]: --	[n22]: 1.5	[n28]: 0.6	[n34]: 1.0	[n40]: 1	[n46]: --	[n54]: ft	[n66]: --
5											[n55]: fmax	[n67]: --
6		[n3]: pnp	[n6]: 0	n11:12]: 0-0.16	[n17:18]: --	[n23:24]: 1.5	[n29:30]: 0.7	[n35:36]: 1.1	[n41:42]: 1	[n47:48]: --	[n56:59]: fmax-ft	[n68:71]: --
7			[n7]: 0.16	[n13]: 0.16	[n19]: --	[n25]: 1.5	[n31]: 0.6	[n37]: 1.0	[n43]: 1	[n49]: --	[n60]: ft	[n72]: --
8											[n61]: fmax	[n73]: --

		Type	Xeb	Xbc		Vce	Vbestart	Vbemax	GummelAC		RFpar	
1	[n1]: --	[n2]: npn	[n4:5]: 0-0.16	[n8:10]: 0-0.16	[n14:16]: --	[n20:22]: 1.5	[n26:28]: 0.6-0.7	[n32:34]: 1.0-1.1	[n38:40]: 1	[n44:46]: --	[n50:55]: fmax-ft	[n62:67]: --
2		[n3]: pnp	[n6:7]: 0-0.16	n11:13]: 0-0.16	[n17:19]: --	[n23:25]: 1.5	[n29:31]: 0.6-0.7	[n35:37]: 1.0-1.1	[n41:43]: 1	[n47:49]: --	[n56:61]: fmax-ft	[n68:73]: --

To unfold nodes:

1. Select the folded nodes or nodes to the left.
2. Choose **Nodes** > **Configuration** > **Unfold**.

This operation unfolds the experiments collapsed at that point. If you fold experiments to the right of this node, the folding will be retained.

To unfold all experiments in the scenario:

► Choose **Scenario** > **Unfold All**.

This operation unfolds all folded experiments and restores the project tree to its original configuration.

Note:

The configuration of folded nodes is stored with the project and is applied the next time you load the project.

5

Design-of-Experiments Wizard and Taguchi Wizard

This chapter describes the Design-of-Experiments (DoE) Wizard and the Taguchi Wizard that are available in Sentaurus Workbench.

Design-of-Experiments (DoE) Wizard

This wizard facilitates the use of tools for DoE available in Sentaurus Workbench. You sequentially specify settings of different factors that determine the type of design to generate.

Step 1: Selecting the Design-of-Experiments Option

In Step 1 (see Figure 41), you choose one of the design options. These design options implicitly relate to the objectives of users for that design. Each option is explained briefly in the lower part of the window.

Figure 41 DoE Wizard - Step 1

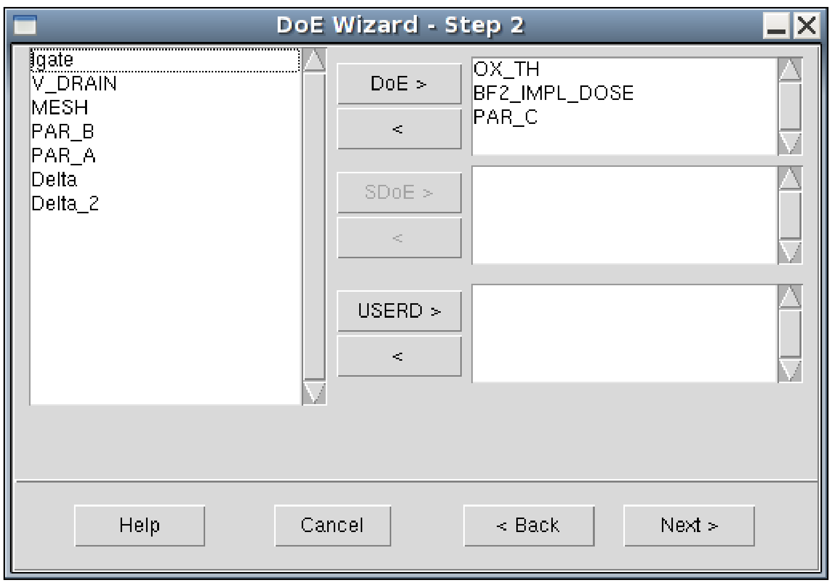
Note:

Although a stochastic design is used to implement an uncertainty analysis instead of an optimization process, this design option is included in the DoE Wizard because an uncertainty analysis can be performed more efficiently with an appropriate stochastic design, such as the one provided by the probabilistic collocation method.

Step 2: Selecting Parameters

In Step 2 (see Figure 42), you select which parameters will be included in the study. The listed parameters in the left pane are those in the current project.

Figure 42 DoE Wizard - Step 2



You must move the parameters under investigation to either:

- The DoE field (upper-right pane).

  Parameters in this field define a DoE suitable to achieve the objective selected in Step 1.

- The SDoE field (middle-right pane).

  The **SDoE** button becomes active only when the **Stochastic Design** option is selected in Step 1, that is, only if the goal is to perform an uncertainty analysis. Therefore, you cannot mix, in the same experimental plan, parameters that by definition have different objectives, such as deterministic parameters that are used to optimize the response and stochastic parameters intended to perform an uncertainty analysis.

- The USERD (user-defined) field (lower-right pane).

  Parameters in this field allow for the previously mentioned DoE to be run under different conditions, with each parameter measuring a different response.

Unselected parameters remaining in the left pane are considered to be constants during the simulation. Their values are those belonging to the experiment selected. If you did not select an experiment when launching the DoE Wizard, their values are the default ones assigned by you when creating the project.

Note:

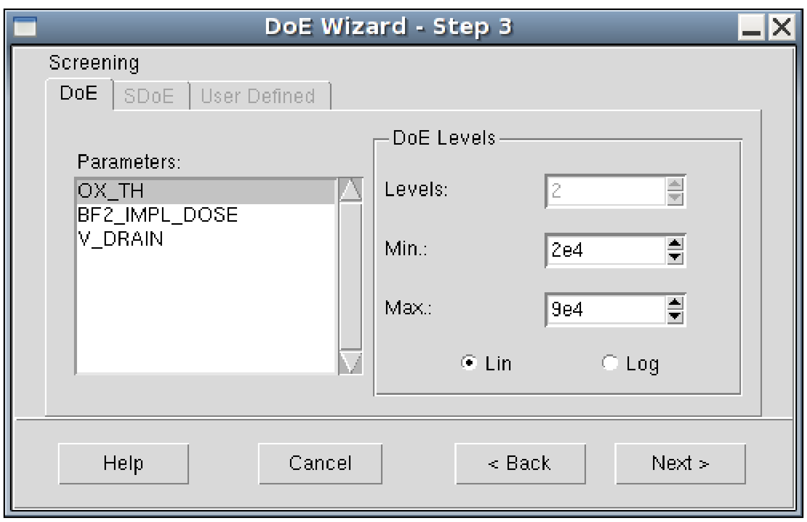
If you selected multiple experiments, the DoE Wizard takes the values of the first selected experiment as constant parameters.

The next steps differ depending on the option selected in Step 1.

Step 3: Screening Option

For a screening process, in Step 3, you must specify the levels for each parameter and the scale in which they are measured (see Figure 43).

Figure 43 DoE Wizard - Step 3 for screening



When you have specified the parameters, Step 4 provides a list of suitable designs to fit a first-order model. The length of the list depends on the number of parameters selected in the DoE field (Step 2).

In Step 4, the Runs column displays the required number of runs to complete each design and the Resolution column shows the resolution of each design (see Figure 44).

You must choose one design. Consider that the higher the resolution, the better the quality of the fitted model will be and the higher the cost of the experimental plan will be.

Figure 44   DoE Wizard - Step 4 for screening

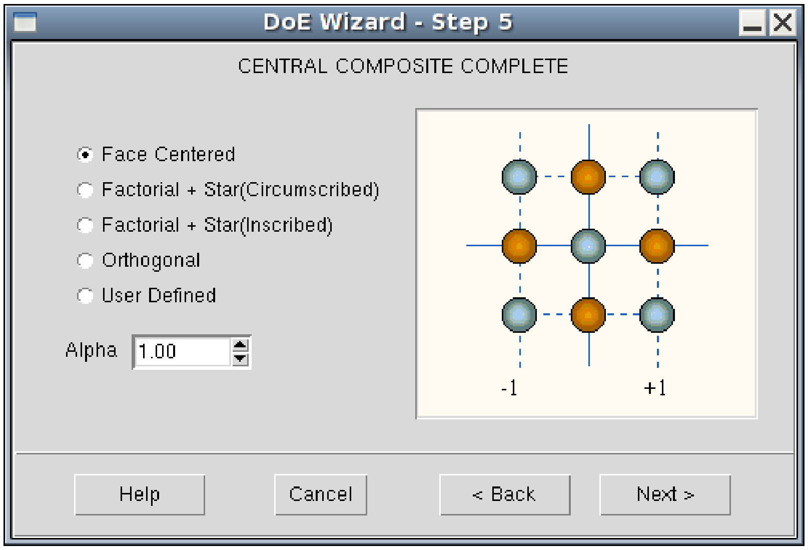
Step 3: Response Surface Model Option

Step 3 for a response surface model (RSM) is the same as for screening (see Figure 43). Nevertheless, you must be aware that, for the **Response Surface Model** option, the objective is to fit a second-order model. This implies that the DoE Wizard adds at least one more level to each parameter automatically.

Step 4 for an RSM provides a list of designs that allow the fitting of a second-order model. The second column in the table shows the number of simulations or runs required to complete each experimental plan.

If Central Composite Complete or Central Composite Small is selected, you must define the axial distance in a special step (see Figure 45).

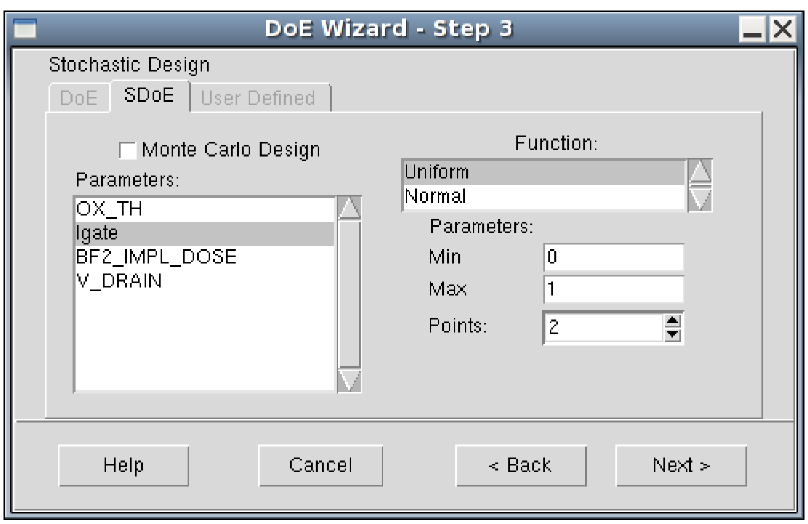
Figure 45 DoE Wizard - Step 5 for response surface model



Step 3: Stochastic Design Option

Step 3 for stochastic design (see Figure 46) allows you to define the characteristics of stochastic parameters. This means that you must associate each parameter with a probabilistic distribution and a number of collocation points (parameter settings).

Figure 46 DoE Wizard - Step 3 for stochastic design

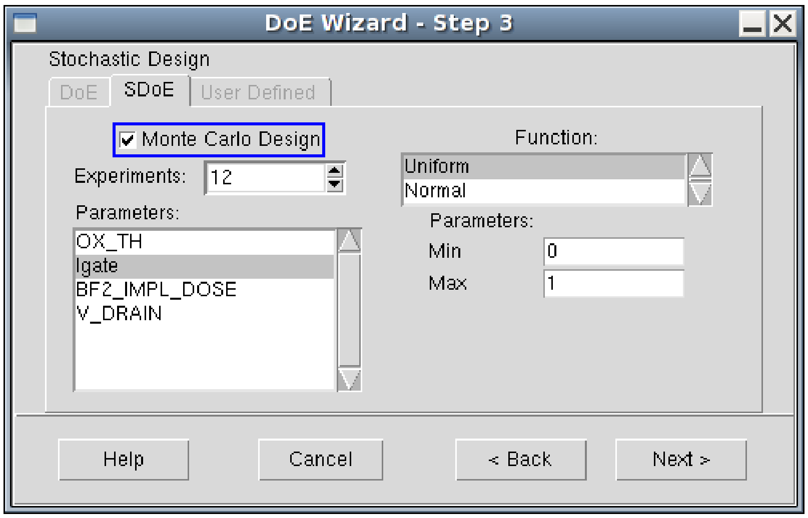


The stochastic design produces experiments by combining random values that are generated according to user settings for each involved parameter: the random distribution and the number of values. The alternative is to generate an independent set of experiments. You can define the final number of experiments to generate, and Sentaurus Workbench creates experiments by generating a random value for each parameter variation.

To use this mode of DoE:

1. Select **Monte Carlo Design** (see Figure 47).
2. In the **Experiments** box, specify the final number of experiments to generate.

Figure 47 DoE Wizard – Step 3 for Monte Carlo stochastic design

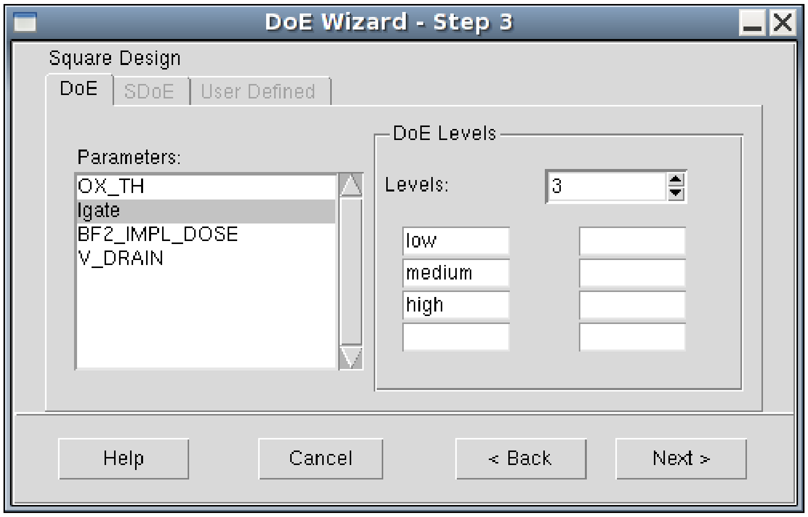


Step 3: Square Design Option

Square designs are deterministic designs that are used to study more than two levels for each parameter effectively, using Latin or Greco Latin or Hyper Greco Latin square.

In this case, in Step 3, you define from three to eight levels for each parameter (see Figure 48).

Figure 48 DoE Wizard - Step 3 for square design



Step 3: Sensitivity Analysis Option

Sensitivity analysis is used to analyze model outputs as a function of very small changes to a single parameter with all other parameters fixed. Therefore, sensitivity analysis reveals only the local gradient of the response surface of the model with regard to a given parameter.

Sentaurus Workbench supports the following sensitivity analysis modes:

- The standard mode provides a powerful and flexible way of generating experiments based on the involved parameters.
- The Taurus™ Workbench–compatibility mode implements the Taurus Workbench style of creating experiments for sensitivity analysis.

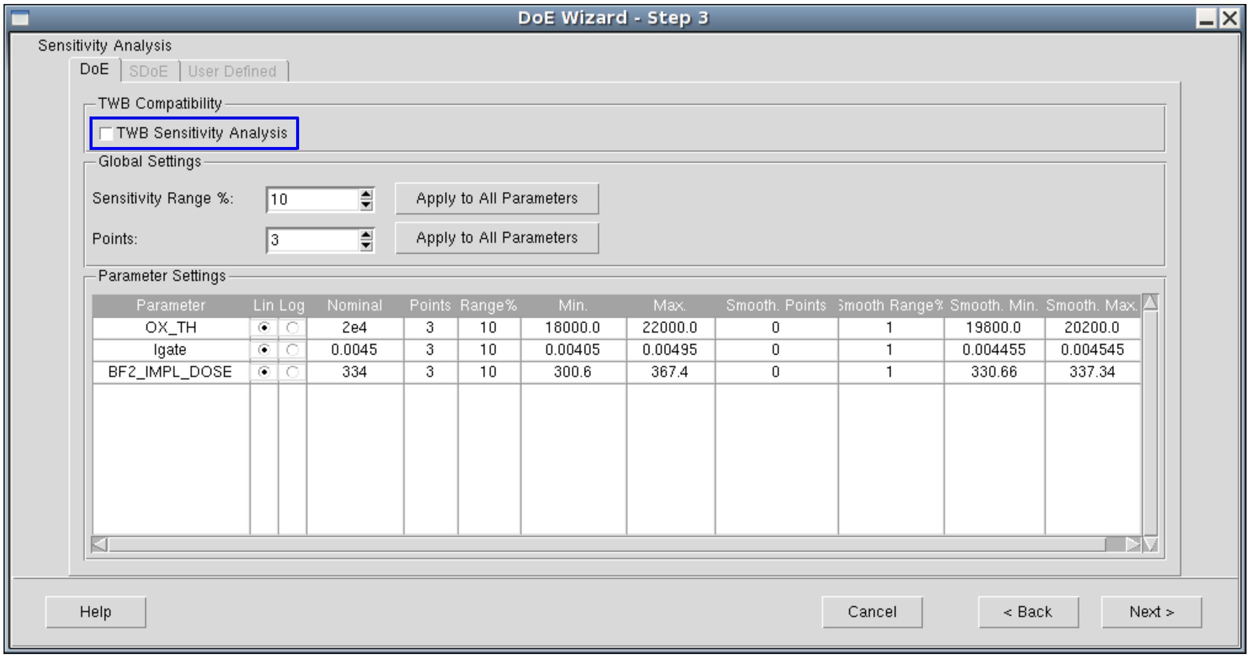
Standard Mode

Step 3 for sensitivity analysis consists of the following group boxes.

TWB Compatibility Group Box

In this group box, you leave the **TWB Sensitivity Analysis** option unselected to use the standard mode (see Figure 49).

Figure 49 DoE Wizard – Step 3 for standard mode of sensitivity analysis



Global Settings Group Box

In this group box, you can specify settings that apply to all of the involved parameters. This can save time when several parameters have similar values of the sensitivity range and the number of samples inside that range.

The **Sensitivity Range** box refers to the range around the nominal value that covers the area of the parameter variations. The default range is 10% and symmetric. For the nominal value *X*, the corresponding sensitivity range is *X*±10%.

The **Points** box specifies the number of samples inside the sensitivity range (the default is 3).

When you click **Apply to All Parameters** for **Sensitivity Range** and **Points**, the global values overwrite the corresponding parameter-specific settings in the Parameter Settings group box.

Parameter Settings Group Box

In this group box, you define settings specifically for each involved parameter.

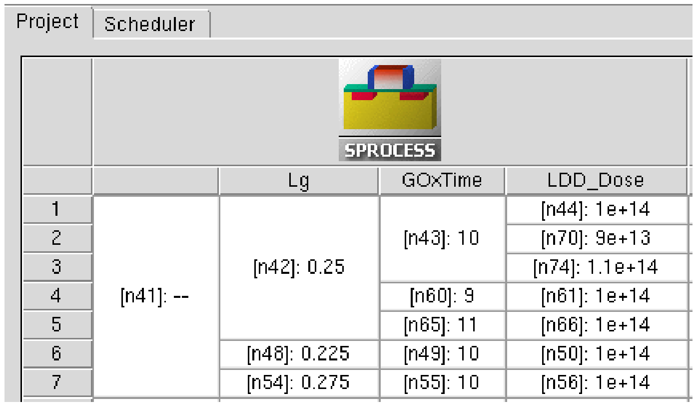
Table 7 Parameter settings

Parameter setting	Description
Lin or Log	Select either linear scale or logarithmic scale.
Nominal	Nominal value of the parameter. By default, this is the value of the selected experiment. Otherwise, it is the default parameter value.
Points	The number of parameter variations (samples).
Range%	The sensitivity range. By default, this is a symmetric range around the nominal value.
Min.	The lower point of the sensitivity range.
Max.	The upper point of the sensitivity range.
Smooth Points	Specifies an additional number of sample points inside the smooth range (the interval around the nominal value). The default is 0. The **Smooth Points** setting can be helpful when the evaluation of the response leads to an unwanted increase or noise at the nominal value of the parameter. In that case, additional points around the parameter nominal value can help to interpolate the response curve (surface).
Smooth Range%	Specifies a smooth range around the nominal value, where additional points (samples) are defined.
Smooth Min.	The lower point of the smooth sensitivity range.
Smooth Max.	The upper point of the smooth sensitivity range.

By default, the symmetric sensitivity range is defined by the parameter nominal value and the value of **Range%** (in percent from the nominal value). You can redefine the default sensitivity range by the direct specification of the **Min.** and **Max.** values. In this case, the **Range%** value is removed to avoid confusion.

After applying the standard mode, Sentaurus Workbench generates the experiments as shown in Figure 50.

Figure 50 Experiments generated by the standard mode of sensitivity analysis



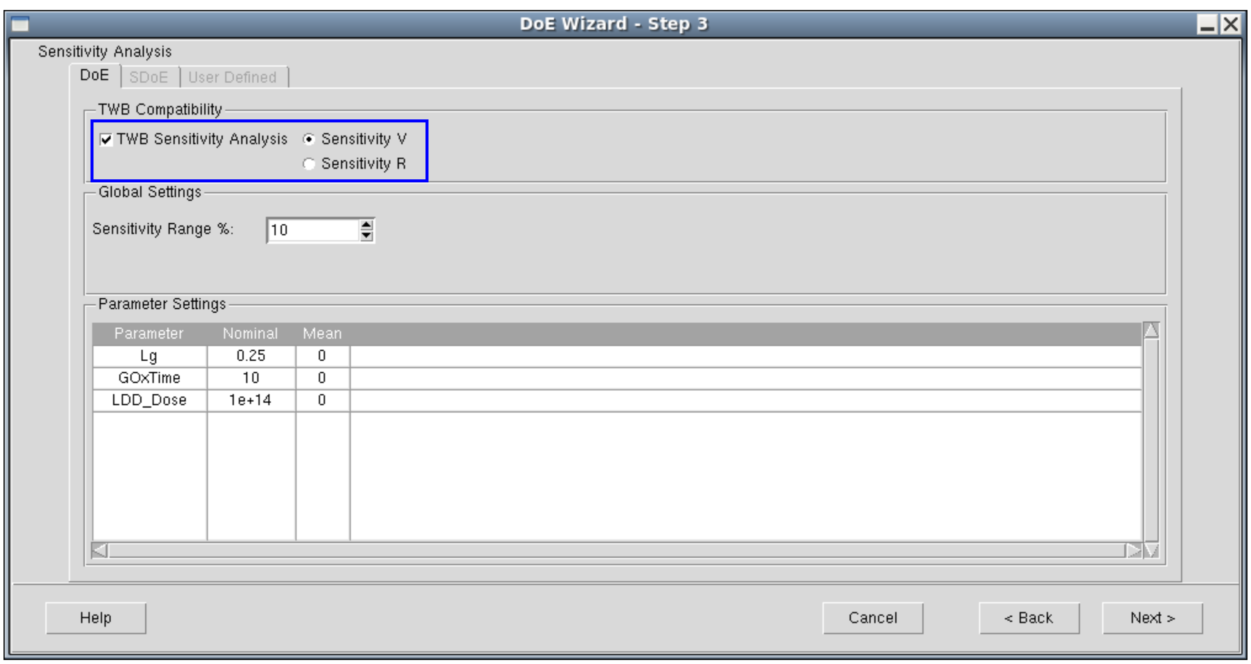
Taurus Workbench–Compatible Mode

Step 3 for sensitivity analysis consists of the following group boxes.

TWB Compatibility Group Box

In this group box, you select the **TWB Sensitivity Analysis** option for the Taurus Workbench–compatibility mode (see Figure 51).

Figure 51 DoE Wizard - Step 3 for Taurus Workbench–compatible mode of sensitivity analysis: Sensitivity V model

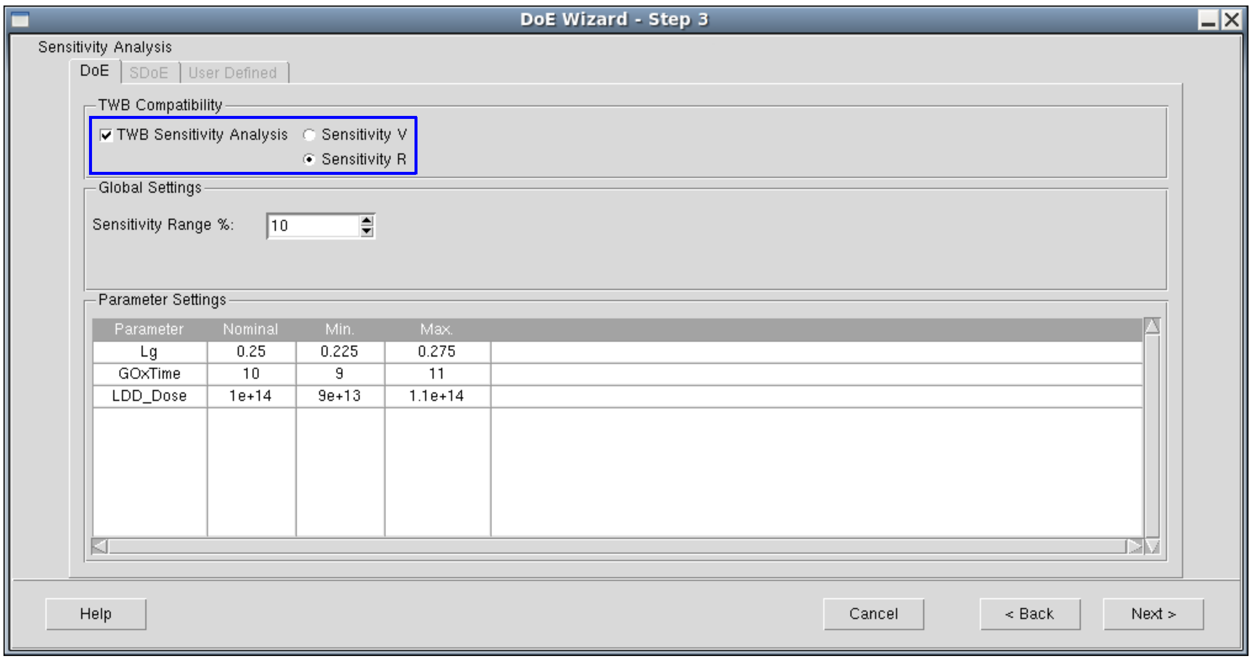


After you select this option, additional options are available to allow you to switch between two models:

- Sensitivity V
- Sensitivity R

The window changes depending on the selected model (see Figure 51 and Figure 52).

Figure 52 DoE Wizard - Step 3 for Taurus Workbench–compatible mode of sensitivity analysis: Sensitivity R model



Global Setting Group Box

In this group box, you specify the sensitivity range, which is a global option for all of the involved parameters. The default is 10%.

Parameter Settings Group Box

In this group box, you can define the following parameter-specific values:

- Nominal
- Minimal and maximal (Sensitivity R model only)
- Mean (Sensitivity V model only)

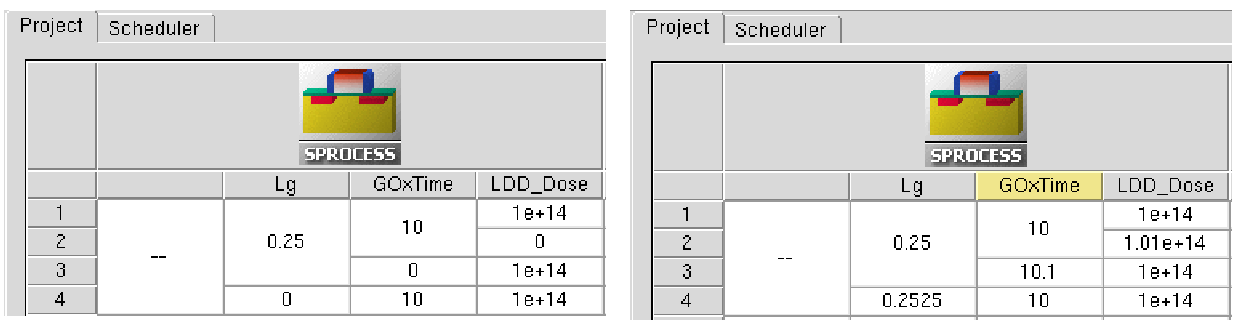
After applying the Sensitivity R model, Sentaurus Workbench generates the following sample values for each involved parameter:

- <nominal>
- (<min + max>) / 2
- (<min> + <max>) / 2 + <sensitivity range> * 0.01 * (<min> + <max>) / 2

After applying the Sensitivity V model, Sentaurus Workbench generates the following sample values for each involved parameter:

- <nominal>
- <mean>
- <mean> + <sensitivity range> * 0.01 * <mean>

Figure 53 Experiments generated by Sentaurus Workbench for (left) Sensitivity V model and (right) Sensitivity R model



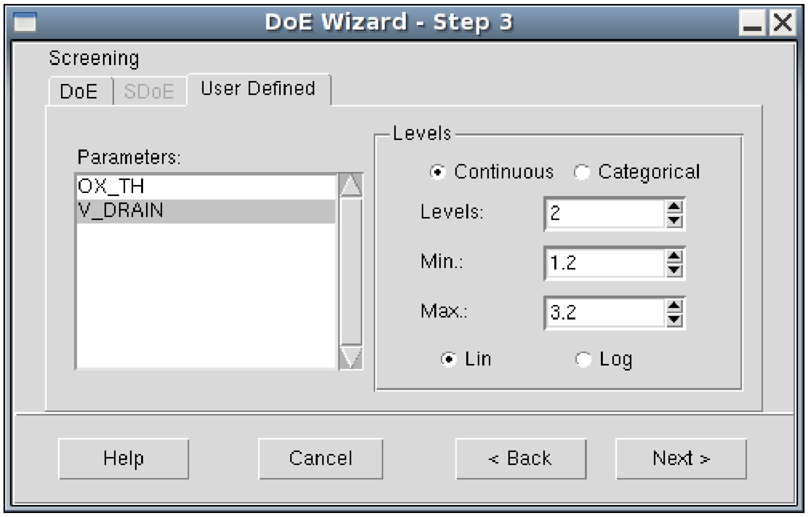
		Lg	GOxTime	LDD_Dose
1	--	0.25	10	1e+14
2				0
3			0	1e+14
4		0	10	1e+14

		Lg	GOxTime	LDD_Dose
1	--	0.25	10	1e+14
2				1.01e+14
3			10.1	1e+14
4		0.2525	10	1e+14

Step 3: User-Defined Parameters

So far, only DoE and SDoE parameters have been explained, but user-defined parameters can be defined in the same way, despite the option selected in Step 1. These parameters can be defined as either continuous with any number of levels or categorical.

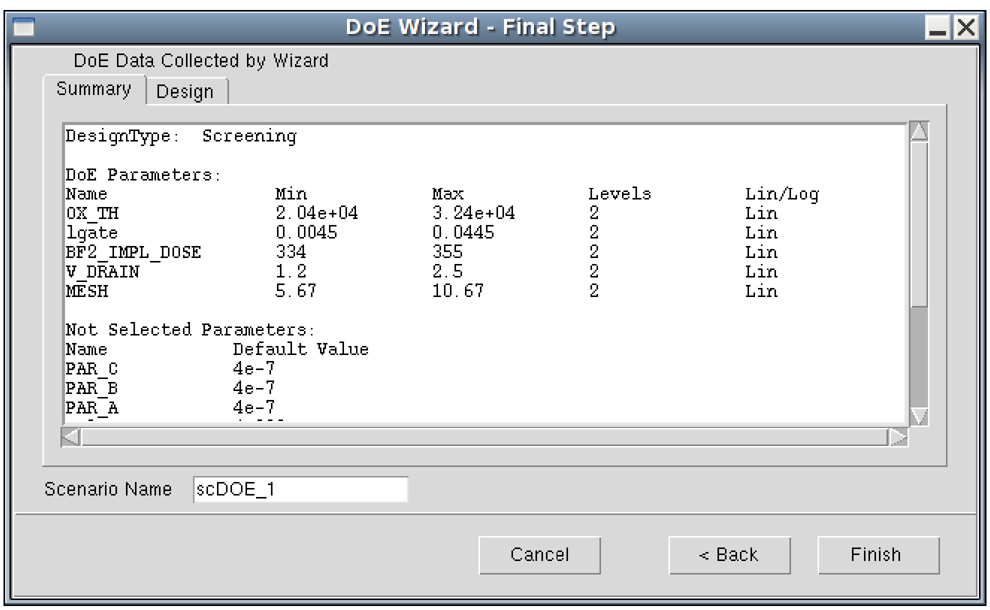
Figure 54 DoE Wizard - Step 3 for user-defined parameters



Final Step: Summary

This step is common to all the design options chosen in Step 1 and summarizes all the relevant information about the design you will generate. It helps you to detect problems such as unfeasible combinations or excessive runs for the resources and time available for the experimental plan.

Figure 55 DoE Wizard - Final Step



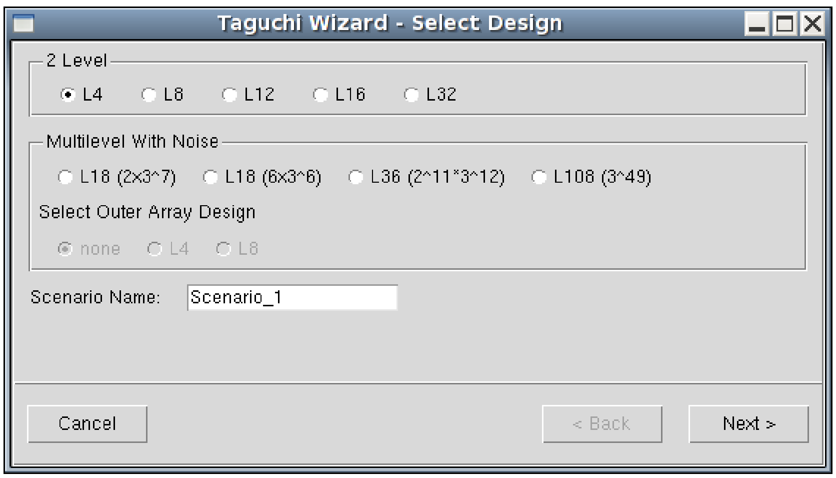
Taguchi Wizard

The Taguchi Wizard helps you to create a Taguchi DoE. This wizard offers basic options that allow for the definition of the characteristics of the required Taguchi design. An optional step shows the values generated according to user specifications in the last steps. The wizard also allows for the generated design to be copied to the main worksheet of Sentaurus Workbench as a new scenario.

Step 1: Selecting the Design

In Step 1 (see Figure 56), you select the number level of the Taguchi design.

Figure 56 Taguchi Wizard - Select Design



Note that:

- Choices are divided between 2 Level and Multilevel With Noise.
- When a Multilevel With Noise design is selected, the **Select Outer Array Design** options are available to allow you to define the design to be used as noise.

Step 2: Specifying the Inner Array

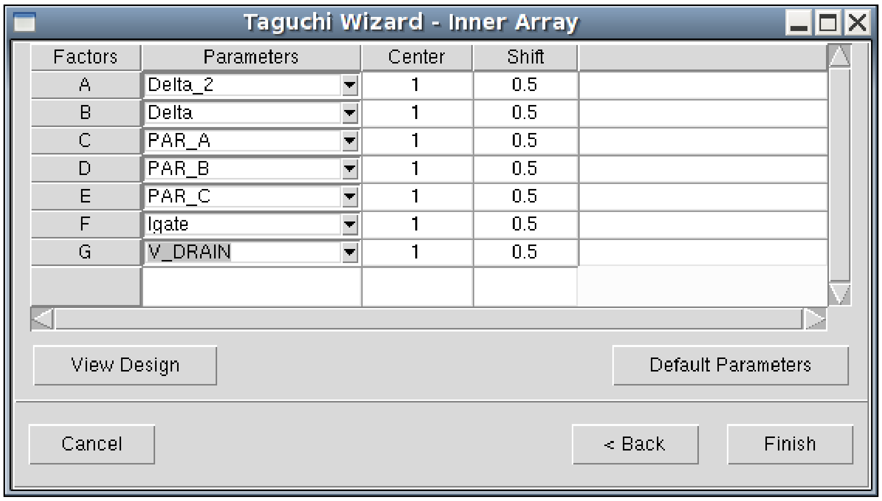
In the next step:

- The Factors column is read only and contains the experiment references designated alphabetically, A, B, C, and so on.

- The Parameters column allows for the selection of one of the valid parameters from Sentaurus Workbench. When a parameter is selected, it is not available for any further selection in the other boxes.

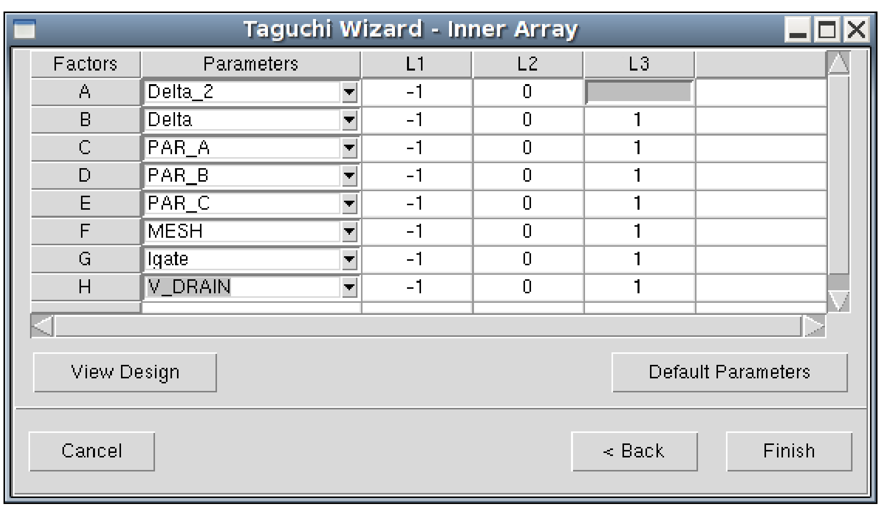
In Step 2 for the 2 Level designs (see Figure 57), you define the design corresponding to the Taguchi inner array. In the Center and Shift columns, you must define parameter values as center ± shift.

Figure 57 Two Level design - Step 2



Step 2 for the Multilevel With Noise designs (see Figure 58) contains entries to select parameter values for each level. In the columns L1 to Ln, you must set the values of the levels for the selected parameter.

Figure 58 Multilevel With Noise - Step 2

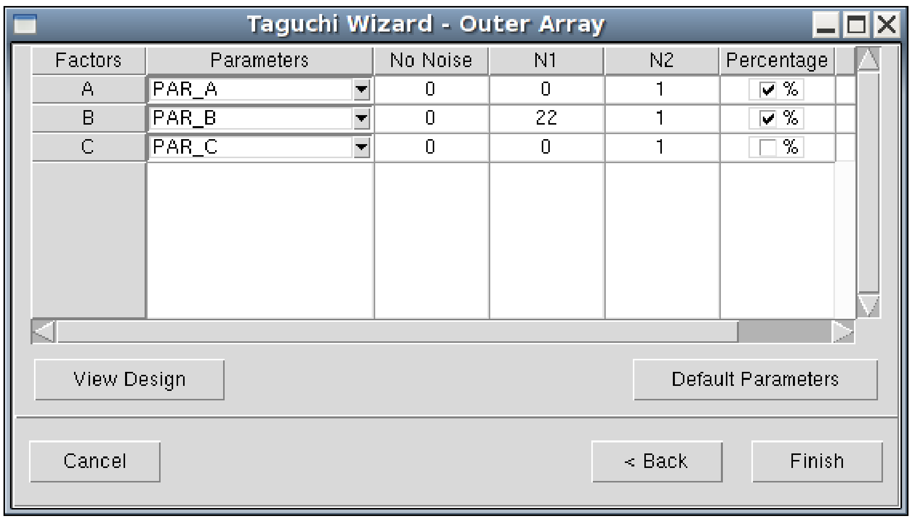


Step 3: Specifying the Outer Array

Step 3 of Multilevel With Noise models (see Figure 59) contains entries to specify the values of the outer (noise) array. The No Noise, N1, and N2 columns allow you to insert three levels of error factor that will be used to create the error design to be applied.

If the Percentage column is selected, the error factor is applied as a percentage of the value (Value With Error = value + value * error / 100). Otherwise, the factor is applied as a simple value (Value With Error = value + error).

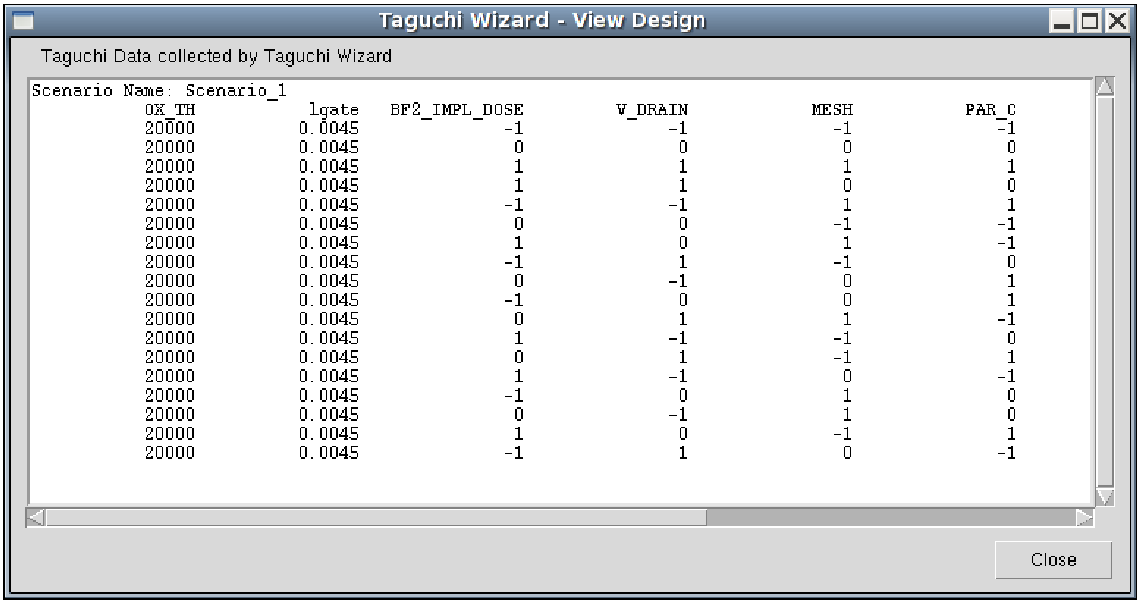
Figure 59 Multilevel With Noise - Step 3



Final Step: Viewing the Design

You can click the **View Design** button (see Figure 59) to display the View Design dialog box (see Figure 60), where you can see the design that will be generated.

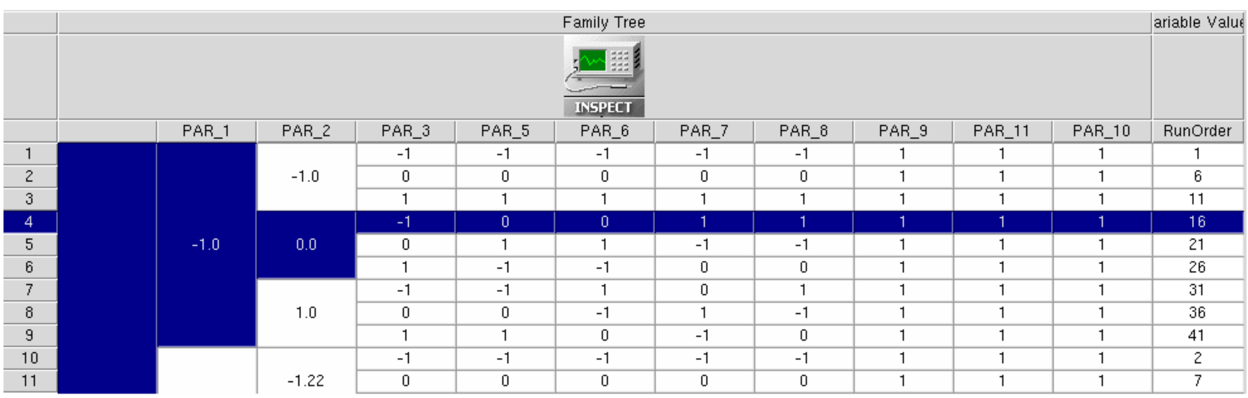
Figure 60 Taguchi Wizard - View Design dialog box



OX_TH	lgate	BF2_IMPL_DOSE	V_DRAIN	MESH	PAR_C
20000	0.0045	-1	-1	-1	-1
20000	0.0045	0	0	0	0
20000	0.0045	1	1	1	1
20000	0.0045	1	1	0	0
20000	0.0045	-1	-1	1	1
20000	0.0045	0	0	-1	-1
20000	0.0045	1	0	1	-1
20000	0.0045	-1	1	-1	0
20000	0.0045	0	-1	0	1
20000	0.0045	-1	0	0	1
20000	0.0045	0	1	1	-1
20000	0.0045	1	-1	-1	0
20000	0.0045	0	1	-1	1
20000	0.0045	1	-1	0	-1
20000	0.0045	-1	0	1	0
20000	0.0045	0	-1	1	0
20000	0.0045	1	0	-1	1
20000	0.0045	-1	1	0	-1

After clicking the **Finish** button (see Figure 59), the selected design is created in Sentaurus Workbench (see Figure 61).

Figure 61 Taguchi Wizard results



		PAR_1	PAR_2	PAR_3	PAR_5	PAR_6	PAR_7	PAR_8	PAR_9	PAR_11	PAR_10	RunOrder
1				-1	-1	-1	-1	-1	1	1	1	1
2			-1.0	0	0	0	0	0	1	1	1	6
3				1	1	1	1	1	1	1	1	11
4				-1	0	0	1	1	1	1	1	16
5		-1.0	0.0	0	1	1	-1	-1	1	1	1	21
6				1	-1	-1	0	0	1	1	1	26
7				-1	-1	1	0	1	1	1	1	31
8			1.0	0	0	-1	1	-1	1	1	1	36
9				1	1	0	-1	0	1	1	1	41
10				-1	-1	-1	-1	-1	1	1	1	2
11			-1.22	0	0	0	0	0	1	1	1	7

6

Preprocessing Projects

This chapter discusses how Sentaurus Workbench preprocesses projects.

Introduction to Project Parameterization

An important feature of Sentaurus Workbench is support for project parameterization. A parameterized project consists of a set of experiments where certain sensible input variables take different values. A parameterized project is represented as a tree structure called `gtree`.

In a project, you provide the set of input file templates for each tool as defined in the tool database. You can express variations in templates using the preprocessor language of Sentaurus Workbench. The categories of preprocessor language constructs are:

- Preprocessor #-commands (see Preprocessor #-Commands on page 163)
- @-references (see @-References and Tree Navigation on page 164)

The Sentaurus Workbench preprocessor also supports special Tcl command blocks, which contain arbitrary Tcl commands (see Using Tcl Command Blocks on page 173).

To differentiate nodes in `gtree` and to take into account already computed results, the Sentaurus Workbench preprocessor (the `spp` utility) automatically generates actual input files at runtime. Dynamic preprocessing allows a simulation to depend on the results of other simulations, that is, the behavior of a node can vary as a function of variables extracted at other nodes.

Global and Runtime Preprocessing

Note:

Locked nodes are not preprocessed.

Preprocessing is performed in two phases:

1. The global projectwide preprocessing phase, before project execution, does not create real input files (`pp*`). Instead, it scans the user input templates of the tool to extract the job interdependencies and to produce the project execution graph (file `gexec.cmd`), finds the `#set` and `#seth` commands, and inserts variable values into the project variable file (`gvars.dat`). In addition, global preprocessing can apply to expressions, nodes, and scenarios (see Node Expressions on page 168).
2. The second phase is the generation of job input files before you start a job. All preprocessor commands and @-references are resolved and replaced at that time. A variable reference is replaced with the most recently extracted value or with the value given at the closest `#set` command if there is no extracted value. Finally, all preprocessor Tcl blocks are extracted, evaluated, and substituted with the result of the Tcl evaluation (standard output).

To perform phase 1 only, use the command:

```
spp PROJECT
```

To perform both phases for the entire project (create the execution graph and variable file, and generate all `pp*` input files), use the command:

```
spp -input PROJECT
```

Note:

In the global preprocessing phase, the variable references are replaced with `#set` or `#seth` values, not with extracted values.

The `gsub` utility automatically performs phase 1 preprocessing when required (if any input file has been modified since the last global preprocessing), before job submission.

The `gjob` utility automatically performs phase 2 preprocessing before job execution, unless the project is in Editable runtime editing mode or the `-nopp` command-line option is specified explicitly.

Preprocessor #-Commands

The #-commands give instructions to the Sentaurus Workbench preprocessor (the `spp` utility). They are used when simple @-references are not sufficient to modify the behavior of a tool. For example, conditional commands instruct `spp` to create different sections of an input file for different nodes (or groups of nodes). Conditions typically refer to parameters, as shown in the following input file fragment of Sentaurus Device:

```
#if @WithHydro@ == 1
coupled { Poisson Electron Hole eTemperature }
#else
```

```
coupled { Poisson Electron Hole }
#endif
```

The following example shows a template with two sections, one for even and one for odd node indices:

```
#if @node:index@ % 2 == 0
# section for even node indices
#else
# section for odd node indices
#endif
```

The following example shows how to test for the next or previous tool:

```
#if "@tool_label|+1@" == "shell2"
next tool is shell2
#else
next tool is something else
#endif
```

See #-Commands on page 301 for a list of all the available #-commands of Sentaurus Workbench.

@-References and Tree Navigation

Since Sentaurus Workbench must control all tool input and output files in the simulation tree and determine job interdependencies, file names must appear only as file references in input files.

Do not use absolute, hard-coded file names. Use node references instead of hard-coded node keys. At runtime, the Sentaurus Workbench preprocessor substitutes file references and node references with the corresponding file names or node keys.

The following conventions and notations are used:

- Each node is uniquely identified by its key noted *nkey*.
- Node keys can be displayed in the Project Editor.
- Tool output file names are prefixed with `n`<nkey>_<acronym>, where <acronym> is a three-character string identifying a specific tool.
- The set of nodes at a certain tree level represents all occurrences of the tool at that level. These nodes are indexed from 1 to n, where 1 denotes the leftmost node and n is the rightmost node. A node index is the absolute position of the node in its tree level.

@-references can be parameters, variables, expressions, or commands that are enclosed in a pair of at signs (@). The preprocessor evaluates and substitutes the enclosed content. A simple reference of Sentaurus Workbench usually refers to the current node (`@node@`).

Navigation operators can be used to move the reference from the current node to another node in the simulation tree. Relative navigation can be in all directions around the current node: up, down, left, right, and also in absolute indices.

To illustrate the use of file references, a typical Sentaurus Device `File` section is:

```
File {
   Grid = "@tdr@"
   Plot = "@tdrdat@"
   Current = "@plot@"
   Output = "@log@
}
```

Note:

For historical reasons, the meaning of the keyword `Plot` is different in Sentaurus Device and Sentaurus Workbench. In Sentaurus Device, `Plot` refers to the plots of primary and derived quantities over the computational domain. In Sentaurus Workbench, `Plot` refers to I–V curves.

@-Expressions

Sentaurus Workbench supports two @-expression types that are evaluated during preprocessing: `@[...]@` and `@<...>@`.

Both types of @-expression can be nested in any way. For example:

```
@< ... @[...]@ ...>@
```

```
@[ ... @<... @[...]@ ...>@ ...]@
```

The following expressions are equivalent:

```
@< @HaloDose@ + 1e-14 >@
```

```
@[ expr {@HaloDose@ + 1e-14} ]@
```

Combined with #-commands, @-expressions help you to make your command files even more flexible.

Expression @[...]@

When the preprocessor finds an expression of `@[...]@` type, it extracts the expression from the brackets and evaluates it in Tcl interpreter like this:

```
% eval <expression>
```

For example, the expression:

```
@[string compare @HaloDose@ 1e-14]@
```

will be processed as follows:

- `@HaloDose@` is substituted with the corresponding Sentaurus Workbench value for the current node.
- The following command is evaluated in the Tcl interpreter:

```
% eval string compare <HaloDoseValue> 1e-14
```

Sentaurus Workbench considers the `@[...]@` expression to be an arbitrary Tcl command. Therefore, you must enclose Sentaurus Workbench parameters and variables with @-characters when referring to them inside an `@[...]@` expression. It instructs Sentaurus Workbench to substitute such a reference with the corresponding value for the current node. Otherwise, Sentaurus Workbench considers such a reference to be a Tcl string value, which leads to either an incorrect preprocessing result or an error.

For example, the following expression is always substituted with 0 as the strings `"HaloDose"` and `"1e-14"` are not equivalent:

```
@[string compare HaloDose 1e-14]@
```

The following expression generates a preprocessing error as the Tcl command `expr` fails on a nonnumeric argument:

```
@[expr HaloDose + 1e-14]@
```

Expression @<...>@

When the preprocessor finds an expression of `@<...>@` type, it extracts the expression from the angle brackets and evaluates it in Tcl interpreter like this:

```
% eval expr { <expression> }
```

For example, the expression:

```
@< @HaloDose@ + 1e-14 >@
```

will be processed as follows:

- `@HaloDose@` is substituted with the corresponding Sentaurus Workbench value for the current node.
- The following command is evaluated in the Tcl interpreter:

```
% eval expr {<HaloDoseValue> + 1e-14}
```

Unlike `@[...]@` expressions, Sentaurus Workbench allows you to use references to parameters and variables without enclosing them with @-characters inside an `@<...>@` expression. The following expressions are equivalent:

```
@< @HaloDose@ + 1e-14 >@
```

```
@< HaloDose + 1e-14 >@
```

Note:

The syntax of the second expression is supported for legacy reasons, namely, to support old simulation projects. Although Sentaurus Workbench supports both syntaxes, you should use the first syntax in new simulation setups.

Redefining Delimiters

Note:

Redefining delimiters works only when the Sentaurus Workbench preprocessor (the `spp` utility) is used.

As mentioned in @-References and Tree Navigation on page 164, the preprocessor recognizes the at sign (@) as a delimiter for parameters, variables, expressions, or commands. At the same time, the input language syntax of some tools integrated into Sentaurus Workbench allows the use of the at sign (@) internally in language-specific constructions, for example, internal variables for Taurus TSUPREM-4.

To avoid problems during preprocessing, all such character expressions in the input file must be protected with the `#verbatim` keyword. However, this can overload the input file with preprocessor commands and instructions, making the file difficult to read. Other preprocessing problems might arise when a preprocessor reference `@...@` is used on the same line as an `@`... language construction.

To avoid such problems, the default delimiter symbol can be redefined for a tool by setting up a particular tool database instruction. For example:

```
set WB_tool(<tool_name>,substitution_delimiter) <delimiter>
```

where `<tool_name>` is the tool name. The delimiter can be either a single character or a combination of characters (space is not allowed).

The at sign (@) remains a valid delimiter for input files of other tools. The redefined delimiter takes effect for all of the tool-related files involved in the preprocessing. At the same time, tool-specific setup, prologue, and epilogue scripts from the tool database, as well as the variable extraction commands, can be specified only using the standard @-delimiter.

Preprocessing Nodes

You can use node filters and node expressions when preprocessing projects.

Node Filters

A node filter is a Tcl expression that evaluates to true (!=0) or false (0) for any existing node in the simulation tree. The syntax of a filter is:

```
filter: "{" EXPR "}"
```

`EXPR` is a Tcl expression that evaluates to a number and can refer to parameters or variables existing in the current tree using the `$NAME` notation. For example, the following expression will evaluate to true for nodes where the value of parameter `P2` is equal to 3 and the value of variable `V1` is greater than 2:

```
{ $P2 == 3 && $V1 > 2 }
```

Nodes for which a referenced parameter or variable is not defined are rejected (evaluated to false). In other words, the previous expression is equivalent to:

```
{ ([info exists P2] && $P2 == 3) && ([info exists V1] && $V1 > 2) }
```

In addition, `EXPR` recognizes the predefined functions `min(NAME)` and `max(NAME)`. The following example will evaluate to true for nodes where the value of parameter `P2` is equal to the minimal value of `P2`:

```
{ $P2 == min(P2) }
```

Node Expressions

Sentaurus Workbench provides expressions – abbreviated to `gexpr` – to selected nodes in the simulation tree. A node expression `gexpr` returns a list of node keys and is used especially in the Scheduler to submit nodes for execution. See Node Expressions on page 304 for the syntax of node expressions.

The following example returns the leaf nodes belonging to both scenarios `sc1` and `sc2`, where parameter `P1` is equal to 1:

```
"sc1|last:{$P1 == 1} * sc2|last:{$P1 == 1}"
```

Examples

Combining node expressions and node filters is a powerful tool for scripting in tool input command files.

The following preprocessor expression is resolved as a list of nodes of the Sentaurus Structure Editor tool (`sde`) where the parameter `Type` takes the value `nMOS`:

```
@node|sde:{$Type == "nMOS"}@
```

In the following example, the preprocessor expression is resolved as a list of nodes of the tool `sde` where the parameter `Type` takes the value `nMOS` and the parameter `lgate` takes a value that is less than `0.050`:

```
@node|sde:{$Type == "nMOS" && $lgate < 0.050}@
```

Split Points

A parameter introduces a new level in the simulation tree to the right of the tool step. Viewed from the tree, a parameter splits the overall tool simulation phase into two subphases. This is only an abstract view because each tool instance usually executes the entire simulation phase from the beginning to end in one run.

As a result, the same first subphase (or subphases) is unnecessarily executed several times. This is only true for tools whose command file defines a linear control flow (that is, a flow executed sequentially from the beginning to end, command after command). These tools are *linear* tools. Typically, process simulation tools are linear tools.

Sentaurus Workbench provides a splitting facility so that linear tools can save and restore their state; each intermediate branch in the tree can be executed only once. This facility can save considerable computing time, especially for lengthy jobs with multiple phases.

A tool can be split only when the `Save` and `Load` commands are defined in the tool database. For example, Sentaurus Process has the following two lines in the standard tool database:

```
set WB_tool(sprocess,split,save_cmd) "struct tdr=n@node@"
set WB_tool(sprocess,split,load_cmd) "init tdr=n@previous@"
```

Note:

To ensure that Sentaurus Workbench shows the real status of intermediate nodes, confirm whether **View** > **Tree Options** > **Check Virtual Nodes** is selected (see Nodes on page 65).

Preprocessing Variables

Like parameters, variables hold a value and can be referenced in any form of @-references. The main difference is that a variable does not create a split in the simulation flow and, therefore, does not change the shape of the simulation tree.

The `#set` command defines a preprocessed variable and assigns a value to it:

```
#set <varname> <value>
```

The `#seth` command is the same as the `#set` command except that the variable is hidden in the Variables Values view in Sentaurus Workbench:

```
#seth <varname> <value>
```

A defined variable can then be referenced by its name in any @-reference notation. A variable reference creates an execution dependency from the node where the reference is performed to the first ancestor where the variable has been set. The scope of a variable is similar to that of a parameter, that is, from the node with the first `#set` or `#seth` directive to all of its descendants.

You can reassign a variable in one of the following ways:

- Use another `#set` or `#seth` directive, that is, override it with another preprocessed value.
- Perform an extraction, that is, override it with an extracted value (see Extracted Variables on page 170).

All variables are shown in the Family Tree.

A typical use of `#set` and `#seth` commands is to define a set of variables bound to a parameter, such as the fields of a record:

```
#if @PARAM@ == 1
#set PARAM_V1 1.5
#seth PARAM_V2 2.5
#elif @PARAM@ == 2
#set PARAM_V1 11.5
#seth PARAM_V2 12.5
#endif
```

This construct avoids the declaration of two artificial parameters and reduces the size of the overall project tree.

Extracted Variables

Another common use of variables is to extract a value from the tool output file and refer to it in the next tool input file.

Values are extracted from node output files. After the node has been successfully executed, its output file is parsed for strings that match the following mask:

```
DOE: <varname> <value>
```

Then, the values found are written to the `gvars.dat` file. If the specified variable already exists, the extracted value will overwrite the existing one. If the specified variable does not exist, that is, it was not defined globally or with the `#set` or `#seth` command, a new variable will be created. After that, the extracted variable can be referenced in any subsequent tool input files by using standard preprocessor @-references.

The only way to inform Sentaurus Workbench that a value must be extracted during the simulation and assigned to some variable is to provide the `DOE: <varname> <value>` string in the node output file. Usually, it should be printed in the tool command file. The corresponding command strictly depends on the tool and its command syntax. For example, for Sentaurus Process, it can be a simple Tcl command:

```
puts "DOE: ENERGY 120.56"
```

In the Inspect command file, you can export values to the Family Tree using the predefined `ft_scalar` function:

```
ft_scalar <varname> <value>
```

This function prints the `DOE` extraction mask to the node output file as you would do manually with the `puts` Tcl command.

Extracted values and preprocessed values are separated. Extracted values become available during the simulation running phase, and preprocessed values are already known at the preprocessing stage.

Note:

A common cause of errors when using extracted variables is that a corresponding variable does not exist during preprocessing. In this case, preprocessing of the project can fail if there are any references to this variable in the input files of other tools. To avoid this error, globally declare the variable or use the `#set` preprocessing command. You can do this with the `#seth` preprocessing command if you want to hide the variable in the Variables Values view in Sentaurus Workbench.

The following example demonstrates how the algorithm extracts values from the simulation step and accesses them in the subsequent simulation steps:

1. In the tool where the extraction is performed, an arbitrary default value must be assigned to the extracted variable:

   ```
   #set EXTRACTED_VAR 0
   ```

   This command can be also specified in any preceding tools if they exist.

2. In the tool where the extraction is performed, the printing command must be specified in the tool command file. In the case of Sentaurus Process, it could be one of the following:

   ```
   puts "DOE: ENERGY 120.56"
   puts "DOE: ENERGY @ENERGY_2@"
   ```

   where `ENERGY_2` is another variable.

3. The variable EXTRACTED_VAR can be referenced in any subsequent tool input files by using standard preprocessor @-references and #-commands. For example:

```
@EXTRACTED_VAR@
@< @EXTRACTED_VAR:+1@ / 2 >@
#if @EXTRACTED_VAR@ > 1.5e15
...
#endif
```

Note:

Extracted variables work only for tools for which the extracting algorithm has been defined in the tool database. For example, the global tool database defines it for Sentaurus Process in this way:

```
set WB_tool(sprocess,epilogue) { extract_vars "$nodedir"
   @stdout@ @node@; ... }
```

where the extract_vars function is defined also in the global tool database. This definition means that the values for each executed node will be extracted from the output file on the basis of the DoE: the pattern specified in the function. In the global tool database, this extraction algorithm is defined for all supported simulation tools and utilities. To enable your own tools to support extracted variables, the corresponding definition must be specified in the user or project tool database. It is also possible to define a user-specific extracting algorithm for extracting values between Family Tree steps.

Execution Dependencies

During preprocessing, all dependencies in the files are analyzed in the following way.

If there is a file reference @file_type@ with possible, additional, relative direction suffixes (see Appendix A on page 295 for information about @-references and tree navigation) in any of the command files or parameterized input files, the execution is made dependent on the successful completion of the pointed node.

Note:

@node@ references do not create any dependencies and there are no implicit child-to-parent dependencies.

Dependencies can be forced using the #setdep command. For example, the following command creates a barrier before the current level, that is, no node at the current level can start execution before all nodes at the previous level have been successfully completed:

```
#setdep @node|-1:all@
```

The Sentaurus Workbench preprocessor (spp) checks for circular references and fails if one is found.

You can view dependencies by extending the selection to prerequisite nodes (see Viewing Node Dependencies on page 128).

You can remove dependencies explicitly using the `#remdep` command. For example:

```
#remdep @node|+1@
```

Note:

Although the reference `@previous@` is, in principle, equivalent to `@node|-1@`, the difference is that `@previous@` creates a dependency, while `@node|-1@` does not.

Setting and Unsetting Dependencies

The order in which dependencies are set or unset is important. Unsetting a dependency does not have any effect if the dependency is reset later in the same input file, either implicitly or explicitly. Therefore, a `#remdep` command at the beginning of a file does not have any effect. This command typically appears at the end of an input file.

Using Tcl Command Blocks

The concept of Tcl command blocks is to bring the power of Tcl to the complex languages used by some tools incorporated in Sentaurus Workbench.

Creating Tcl Command Blocks

Tcl command blocks consist of an arbitrary set of Tcl command lines that are delimited by `"!("` and `")!"`. During the final stage of preprocessing, the preprocessor extracts and evaluates Tcl command blocks. In the node input file, each Tcl block is replaced with the standard output of its Tcl evaluation.

The following example illustrates the use of Tcl command blocks to power some Sentaurus Device code. Consider the following part of an input file for Sentaurus Device (Tcl blocks are italicized):

```
#if @<polarization>@ == "on"
!(
#===============================================================
# Computation of piezo/spontaneous charge
# using SWB capability to interpret Tcl command blocks
#===============================================================
set q 1.602e-19; # elementary charge
set Psp_AlN [expr -8.1e-6/$q]; # AlN build-in spontaneous
polarization
set Psp_GaN [expr -2.9e-6/$q]; # GaN build-in spontaneous polarization
set Psp_AlGaN [expr @x@*$Psp_AlN+(1-@x@)*$Psp_GaN]; # AlGaN spontaneous
```

```
polarization
set DPsp [expr $Psp_GaN-$Psp_AlGaN]; # interface charge due to
spontaneous polarization
set e33i [expr (@x@*1.46e-4+(1-@x@)*0.73e-4)/$q]
set e31i [expr (@x@*-0.60e-4+(1-@x@)*-0.49e-4)/$q]
set c13i [expr @x@*108+(1-@x@)*103]
set c33i [expr @x@*373+(1-@x@)*405]
set straini [expr @x@*(3.189-3.112)/(@x@*3.112+(1-@x@)*3.189)]
set Ppz_AlGaN [expr 2*$straini*($e31i-$c13i/$c33i*$e33i)]
set DPpz $Ppz_AlGaN
set intCharge [expr $DPsp+$DPpz]; # resulting value of the interface
charge
# Transfer the resulting charge to the project Family Tree
# as SWB variable
set SWB_VARIABLES(Charge) [format %.6e $intCharge]
)!
* Spontaneous polarization for AlGaN: !(puts  -nonewline [format
%.2e $Psp_AlGaN])!
* Piezopolarization for AlGaN: !(puts  -nonewline [format %.2e
$Ppz_AlGaN])!
* Total AlGaN Polarization: !(puts [format %.2e [expr
$Psp_AlGaN+$Ppz_AlGaN]])!
* Total GaN Polarization: !(puts [format %.2e $Psp_GaN])!
#endif
...
Physics(materialinterface="GaN/AlGaN")
{
#if @<polarization>@ == "on"
   Charge( Conc=!(puts  -nonewline [format %.4e $intCharge])! )
#endif
#if [string match "*NL*" "@model@"]
* hetero barrier non-local barrier tunneling
   Recombination(eBarrierTunneling(nonlocal))
#endif
}
...
```

This file is preprocessed to the following node file:

```
* Spontaneous polarization for AlGaN: -5.06e+13
* Piezopolarization for AlGaN: -1.85e+13
* Total AlGaN Polarization: -6.91e+13
* Total GaN Polarization: -1.81e+13
...
Physics(materialinterface="GaN/AlGaN")
{
   Charge( Conc=1.3925e+13 )

* hetero barrier non-local barrier tunneling
  Recombination(eBarrierTunneling(nonlocal))
}
...
```

Note:

If a Tcl command block prints nothing (no Tcl `puts` command is used), the standard output of the Tcl evaluation is empty. This is the reason why there will be nothing from this block in the preprocessed node input file.

Preprocessing Tcl Command Blocks

The Sentaurus Workbench preprocessor evaluates Tcl command blocks automatically at the final stage of project preprocessing. After the evaluation of preprocessor #-commands, @-references, and @-expressions is completed, Sentaurus Workbench starts evaluating Tcl command blocks (see Preprocessor #-Commands on page 163, @-References and Tree Navigation on page 164, and @-Expressions on page 165).

In the node input file, each Tcl command block is replaced with the standard output of its Tcl evaluation. Evaluation of Tcl command blocks follows the experiment-wise rule: Tcl blocks appearing in the input files of the nodes belonging to the same experiment are evaluated in the context of the same Tcl interpreter, from the first node to the last node. This rule makes it possible to organize an experiment-wise logic of Tcl command blocks that appear in the nodes of the same experiment. For example, a Tcl variable that you set in the Tcl command block of the first node in the experiment is accessible in Tcl command blocks appearing in all other nodes of the same experiment.

Tcl command blocks cannot be used in preprocessor #-commands because Tcl blocks are evaluated only at the final stage of project preprocessing. As the result, the evaluation of #-commands fails.

For example, using Tcl blocks in an `#include` preprocessor command is incorrect:

```
#include "!(puts -nonewline "[file join /the/path myfile.cmd]")!"
```

In such a case, use an @-expression instead:

```
#include "@[file join /the/ myfile.cmd]@"
```

To reduce the preprocessing time, you can call Tcl preprocessing explicitly in one of the following ways:

- Choose **Project** > **Operations** > **Preprocess Tcl Blocks**.
- Press Ctrl+B.
- Right-click a project and choose **Project** > **Preprocess Tcl Blocks**.

In this preprocessing mode, only Tcl command blocks in the tool input files are Tcl evaluated. All unresolved Sentaurus Workbench parameters and expressions (`@...@`, `@<...>@`, `@[...]@`) are substituted with dummy values. This light preprocessing mode is useful for testing purposes only. To obtain the expected results, the entire preprocessing procedure must be activated.

Tcl Command Blocks and Sentaurus Workbench Variables

Tcl command blocks provide a possibility to create or update Sentaurus Workbench variables. They are equivalent to preprocessing variables created with the `#set` and `#seth` commands. For this purpose, a special `SWB_VARIABLES` Tcl array must be updated in the Tcl command block. This array is global.

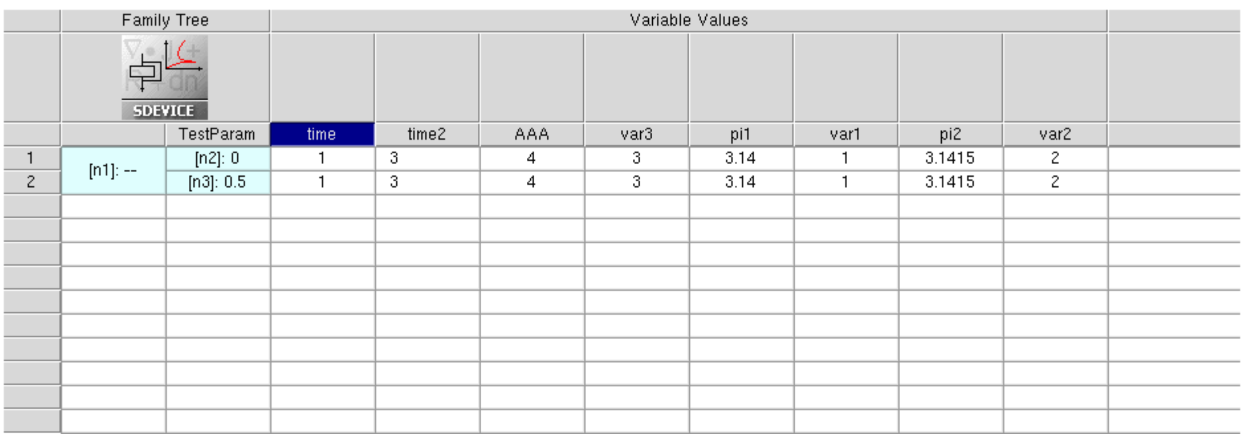
To instruct the Sentaurus Workbench preprocessor to initialize the Sentaurus Workbench variable `"myvar"` with the value `"myval"`, the following Tcl instruction must be inserted into the Tcl command block:

```
set SWB_VARIABLES(myvar) "myval"
```

The following example creates the variables `var1`, `var2`, and `var3` with the values 1, 2, and 3:

```
!(
   ...
   set SWB_VARIABLES(var1) 1
   set SWB_VARIABLES(var2) 2
   set SWB_VARIABLES(var3) 3
   ...
)
```

Figure 62 Creation of three variables



	Family Tree		Variable Values								
	SDEVICE										
		TestParam	time	time2	AAA	var3	pi1	var1	pi2	var2	
1	[n1]: --	[n2]: 0	1	3	4	3	3.14	1	3.1415	2	
2		[n3]: 0.5	1	3	4	3	3.14	1	3.1415	2	

Input and Output Operations Inside Tcl Command Blocks

It is necessary to force flushing of output streams in Tcl command blocks. Use the Tcl `flush` command when writing files inside a Tcl command block. Otherwise, the file will not be available until the node has been executed.

For example:

```
set FID [open "@pwd@/tmp_n@node@_ins.cmd" w]
...
...
puts $FID "Hello World"
flush $FID
close $FID
```

When to Use Tcl Command Blocks

Using Tcl command blocks is helpful in the following circumstances:

- For comprehensive calculations that are difficult to implement in the language of the tool and that use the result value in the native language constructions of the tool. For example, in the case of Sentaurus Device, you could implement in the `sdevice_des.cmd` file:

```
!(
if { "@Type@" == "nMOS" } {
   set SIGN 1.0
   set HFS1 "eHighFieldSaturation( CarrierTempDrive )"
   set HFS2 "hHighFieldSaturation( GradQuasiFermi )"
   set DG "eQuantumPotential"
   set cTemp "eTemperature"
   set EQN0 "Poisson eQuantumPotential Electron Hole"
   set EQNS "Poisson eQuantumPotential Electron Hole Temperature
   eTemperature"
} else {
   set SIGN -1.0
   set HFS1 "hHighFieldSaturation( CarrierTempDrive )"
   set HFS2 "eHighFieldSaturation( GradQuasiFermi )"
   set DG "hQuantumPotential"
   set cTemp "hTemperature"
   set EQN0 "Poisson hQuantumPotential Electron Hole"
   set EQNS "Poisson hQuantumPotential Electron Hole Temperature
   hTemperature"
}
)!
```

- For enhancing the capabilities of native tool languages. For example, the following Tcl command block will write 100 Sentaurus Device `Physics` sections with different concentrations in the preprocessed command file of Sentaurus Device (imitation of for-loop):

```
!(
for {set i 0} {$i < 100} {incr i} {
puts "Physics (materialInterface=\"Silicon/Oxide\")"
puts "Charge(Conc=[expr 6.0e11 + $i*1e11])"
}
)!
```

Note:

Using Tcl command blocks for Tcl-aware tools such as Sentaurus Process is allowed but provides no additional benefits. Such a practice might lead to confusion since Tcl command blocks and the native Tcl flow are evaluated on different scopes. In addition, it makes the tool command file less readable.

Summary of Rules for Using Tcl Command Blocks

When using Tcl command blocks, note the following requirements:

- A Tcl block contains an arbitrary number of native Tcl instructions between a `"!("` and `")!"` pair of symbols.
- Tcl blocks cannot be nested.
- A Tcl block can be inserted in any place of any preprocessed file, such as tool command files and parameter files for Sentaurus Device.
- Each file can contain an arbitrary number of Tcl blocks.
- All Tcl blocks are evaluated by a separate Tcl interpreter, which is specific for each experiment in the project.
- All Tcl blocks are evaluated for each tool, from left to right, and from the beginning to the end of each file.
- A Tcl block can contain a reference to the Tcl variable set in a previous block, which can be placed in the same file or another file of the same experiment.
- Sentaurus Workbench replaces each Tcl block with the result of its output (standard output). Therefore, to extract the value of the Tcl variable `"aaa"`, you must specify `"puts $aaa"`. This provides transparency when interpreting Tcl blocks; all blocks are evaluated in the same way. If a Tcl block does not provide any output, nothing will go to the preprocessed file. However, all the Tcl variables and procedures declared in that block, of course, will exist in the interpreter of the current experiment and could be used further.
- The Sentaurus Workbench preprocessor evaluates all the Tcl blocks after the standard preprocessing of `#...` commands and the resolving of `@...@`, `@<...>@`, and `@[...]@` expressions. This means that you can use any of these standard preprocessing directives inside the Tcl blocks. However, you cannot use Tcl blocks in preprocessor #-commands and @-expressions.
- Sentaurus Workbench has a preprocessing mode (choose **Project** > **Operations** > **Preprocess Tcl Blocks**) that evaluates only Tcl command blocks without standard Sentaurus Workbench preprocessing. This preprocessing mode is reasonably fast and can be used for checking purposes.

- Sentaurus Workbench provides a way to show the values of the Tcl variables in the Sentaurus Workbench variables table. The Tcl block must contain a special Tcl command such as:

  ```
  set out ... (comprehensive calculations)
  set SWB_VARIABLES(myvar) $out
  ```

  After the successful evaluation of a Tcl command block, you can see the new Sentaurus Workbench variable "myvar" in the Sentaurus Workbench variables table. This feature works like the standard Sentaurus Workbench #set <varname> <value> and #seth <varname> <value> commands.

7

Running Projects

This chapter discusses how to run projects in Sentaurus Workbench.

Projects can be run from the Project Editor, the command line, or the Scheduler.

Running Projects From the Project Editor

To run a project:

1. Browse the projects on the tree.
2. Open the project in the Project Editor.
3. Select the nodes or open the required scenario.

You can run multiple selected projects from the projects browser.

To run the selected nodes:

1. Select nodes (see Selecting Nodes With Mouse and Keyboard Operations on page 67).
2. Choose **Project** > **Operations** > **Run**, or press Ctrl+R, or click the **Run** button.

Note:

If you want to run all remaining nodes in the project, you do not need to select these nodes manually. Instead, simply launch a project as usual.

To run a scenario:

1. Select the required scenario (default is **all**).
2. Select the nodes to run from either **remaining** (all nodes that do not have the status *done*) or **all**.

   By default, Sentaurus Workbench submits remaining nodes.
3. Choose **Project** > **Operations** > **Run**, or press Ctrl+R, or click the **Run** button.

The Run Project dialog box opens and nodes are assigned, based on the queue definitions and tool assignments of the user. The **Run** button submits the nodes to the corresponding scheduler or queues.

Running Projects From the Command Line

You can submit jobs to queues or launch a specific job.

Submitting Jobs to Queues

The `gsub` utility is the actual backend system, which performs the following:

- Preprocesses a project (on demand).
- Reads queue assignments.
- Expands the generic expression to determine the node or takes the nodes from the list provided.
- Submits jobs to the queues, calls `gjob` on several machines, and executes the jobs.

The syntax of the `gsub` command is:

```
gsub [options] (FILENAME | PROJECT)
```

Table 8 Command-line options of the gsub command

Option	Description
`-e[xpr] "GEXPR"`	Nodes resulting from the `GEXPR` expression
`-h[elp]`	Displays help information
`-max[Experiments] <number>`	Defines the maximum number of concurrently running experiments
`-n[odes] [<scenario> | "<list of nodes>"]`	Remaining nodes in `<scenario>`, or given node numbers
`-q[ueue] "queue name"`	Submits all the jobs to a queue
`-startTime <datetime>`	Start date and time of the submission (default: immediately); format is: `"mm/dd/yyyy hh:mm:ss AM|PM"`
`-v[ersion]`	Displays version information

Table 8 Command-line options of the gsub command (Continued)

Option	Description
`-verbose`	Displays additional initialization and loading information
Arguments	
`FILENAME`	Text file defining a list of jobs
`PROJECT`	Project directory

Example

```
gsub -verbose -e all -q local:default @STDB@/folder/project
```

This command launches all nodes of the given project locally.

Launching a Specific Job

The `gjob` utility runs a given job or the node of a project locally. It performs the following:

- Preprocesses the node of the given project.
- Runs the given job or the node of the given project locally.

The syntax of the gjob command is:

```
gjob [options] -job NAME (FILENAME | PROJECT)
```

Table 9 Command-line options of the gjob command

Option	Description
`-h[elp]`	Displays help information.
`-nice NUMBER`	Job scheduling priority (default: 0): • Highest: -20 • Lowest: 19
`-nopp`	Does not overwrite node input files of the project while preprocessing (only valid if the project is in the Editable runtime editing mode).
`-p[ack]`	Does not generate empty lines while preprocessing.
`-v[ersion]`	Displays version information.

Table 9 Command-line options of the gjob command (Continued)

Option	Description
`-verbose`	Displays additional initialization and runtime information.
Arguments (one of the following)	
`-j[ob] NAME`	Job name or node number.
`FILENAME`	Text file defining jobs.
`PROJECT`	Project directory.

Note:

The `gjob` command cannot use the node-queue or tool-queue assignments as the `gsub` command does. The `gjob` command essentially runs the given node locally.

Example

```
gjob -verbose -job 2 -nopp -nice -10 @STDB@/folder/project
```

This command launches node 2 of the given project locally with a priority of `-10`, and it instructs Sentaurus Workbench to not overwrite the node input files.

Running Projects From the Scheduler

Sentaurus Workbench submits nodes in the order that is defined by implicit and explicit node dependencies of the project:

- Implicit node dependencies come from the tool input or output interface specified in the tool database.
- Explicit node dependencies are usually those you define in the tool input files using `#setdep` or `@previous@` preprocessor instructions and references.

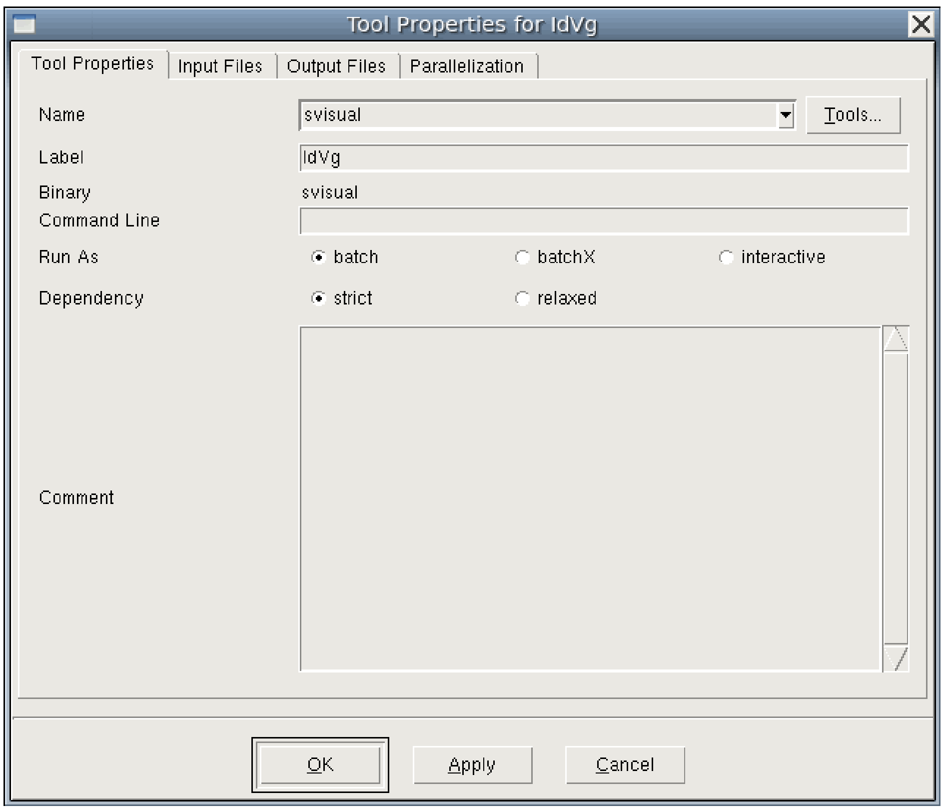
The set of these node dependencies defines the submission order of project nodes. Dependencies can be viewed by extending the selection to prerequisite nodes (see Viewing Node Dependencies on page 128).

Sentaurus Workbench implements a strict dependency model. A node is launched only when all its prerequisite nodes have been executed successfully and appear with the status *done*. If the execution of, at least, one prerequisite node fails or is terminated, Sentaurus Workbench does not launch the next node and removes this node from the execution list.

There can be exceptions to this strict dependency model. In certain cases, you might want to execute the entire simulation project in one run despite some failed nodes. Assume you run a large design-of-experiments (DoE) project and you want to build a model based on this DoE. To build the model, all of the project nodes must be executed. However, some Sentaurus Device nodes might fail because of convergence problems or because your device is punch-through. Despite failures, Sentaurus Device might have already stored important results such as V_{tlin} and V_{tsat}. Then, you want the corresponding Sentaurus Visual or Inspect extraction step to execute and extract these results.

Sentaurus Workbench allows you to relax the strict dependency model for Sentaurus Visual, Inspect, `gtclsh`, `cshell`, and `bash`. These tools are used typically for extracting and visualizing the results produced by Sentaurus Device simulations, so they can complete their tasks even though a Sentaurus Device node failed or was terminated.

Figure 63 Dependency control in the Tool Properties dialog box for Sentaurus Visual



You relax the strict dependency model in the Tool Properties dialog box (see Figure 63). The options in the Tool Properties dialog box for the **Dependency** field are:

- **strict** (default): Sentaurus Workbench launches a node only when all its prerequisite nodes have the status *done*.

- **relaxed**: Sentaurus Workbench launches a node even when its prerequisite nodes have the status *done*, *failed*, or *aborted*.

Runtime Editing Modes for Projects

Projects can have different runtime editing modes.

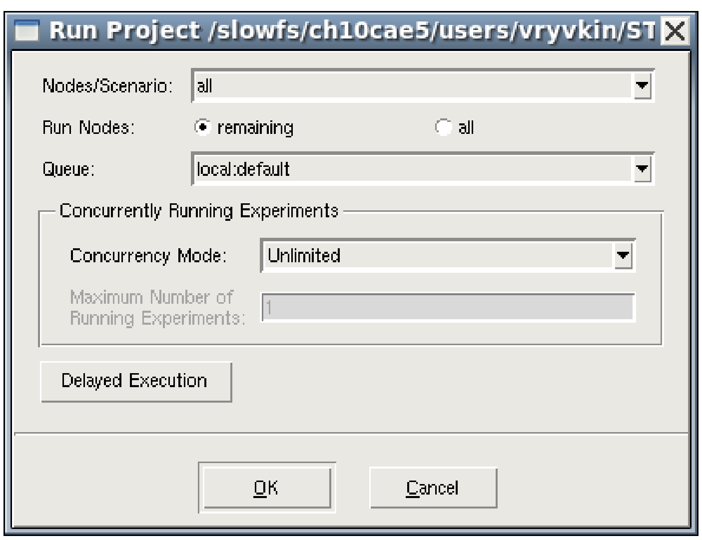
Locked Runtime Editing Mode

As soon as a project is running, it is locked for any changes in the parameterization table. All you can do is run additional nodes, or terminate selected nodes or the entire running project.

Sentaurus Workbench implements strict safeguards to prevent any kind of inconsistency between input project data and the simulation results. Changes made to the tool database files and the tool input files result in the failure of currently running nodes as well as the failure to run pending nodes due to background file timestamp checks.

Runtime preprocessing is a mandatory part of the execution of a project with the Locked mode. All your changes made on the node input files will be lost during the execution; while each node input file will be replaced with the preprocessed tool input file.

Figure 64 Run Project dialog box for a project in Locked mode



Sentaurus Workbench provides a direct and consistent way of scheduling project nodes. The node can be submitted for execution only when the nodes on which it depends (prerequisite nodes) are executed successfully. Sentaurus Workbench automatically determines a list of all the *prerequisite* nodes from the nodes that you select for submission.

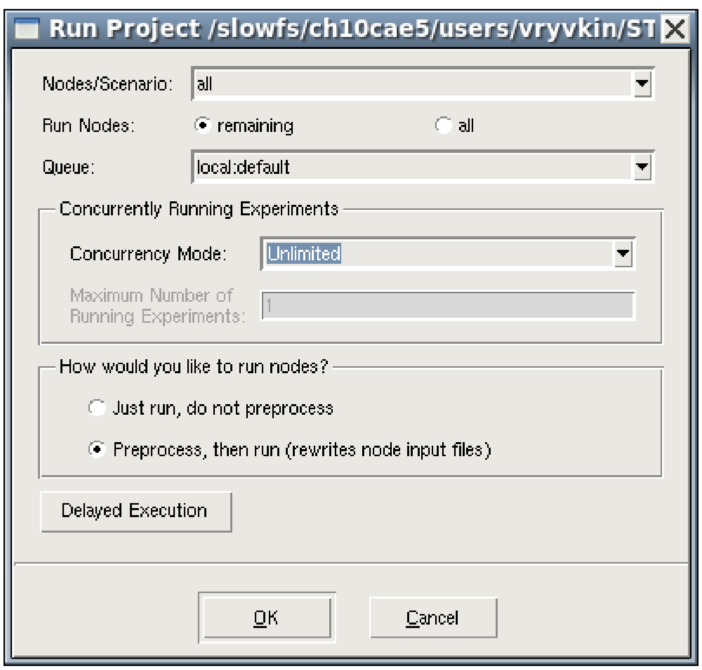
All you need to do is select the nodes to run, and Sentaurus Workbench will handle what is required for this.

Editable Runtime Editing Mode

In contrast to projects in Locked mode, a running project in Editable mode is not locked for changes. You can add or remove tools, parameters, and experiments, and edit tool input files and Sentaurus Workbench configuration files, and so on.

In addition, you can edit node input files and rerun them without the obligatory runtime preprocessing and node file regeneration. To do this, select **Just run, do not preprocess** in the Run Project dialog box (see Figure 65). Changes you made in the node input files are not overwritten by the runtime preprocessing. However, global preprocessing still occurs in this mode. Therefore, if global preprocessing of a node results in an error, then Sentaurus Workbench will not proceed with the execution of your node input file even though it exists.

Figure 65 Run Project dialog box for a project in Editable mode



However, Sentaurus Workbench does not provide background checks of file timestamps in Editable mode. This might result in inconsistency between the tool input file and the simulation result of a certain node. To prevent this, you can preprocess certain nodes or the entire project at any time.

Note:

Preprocessing the entire project is highly recommended after making changes to the parameterization table.

A node can be executed when some prerequisite nodes are not yet completed, have not started, or have failed. When submitting a group of nodes, the order of the node execution takes into account node dependencies inside the group, but it ignores external node dependencies.

Choosing the Appropriate Runtime Editing Mode for Projects

Sentaurus Workbench creates new projects using the runtime editing mode specified in the preferences (see Changing the Default Runtime Editing Mode of Projects on page 33). The Editable mode is the default.

Changing the runtime editing mode of an existing project must be performed manually (see Changing the Runtime Editing Mode of a Project on page 32).

You cannot change the runtime editing mode of a running project, which is why it is highly recommended to specify the appropriate mode for your project *before* you launch it. If you regularly work with typical simulation setups, it is worthwhile making a reasonable choice for the default runtime editing mode for your new projects in the preferences.

For many use cases where you need to change parts of your simulation project and to run them while other project parts keep running, the Editable mode can be the best choice. It gives you maximum flexibility.

The properly chosen runtime editing mode for a project saves time when you create and run your simulations in Sentaurus Workbench.

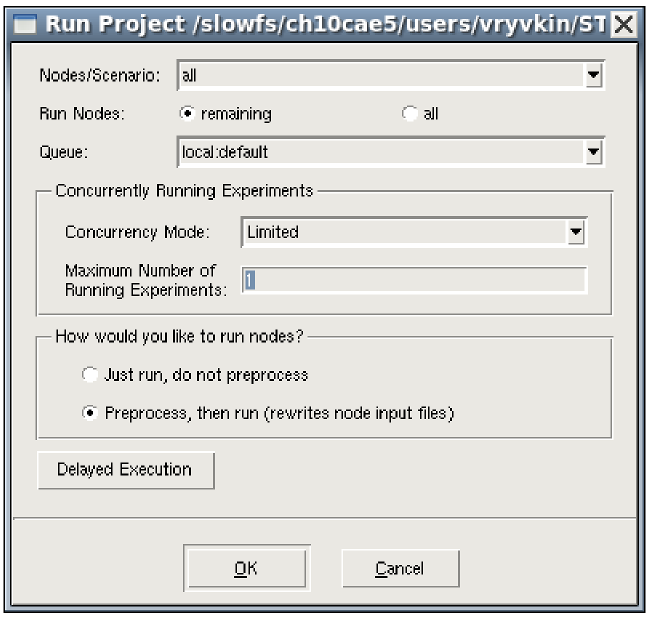
Concurrency Mode for Experiments

The order in which projects are scheduled for execution is defined by node dependencies (see Running Projects From the Scheduler on page 183). This approach guarantees the most effective project execution because Sentaurus Workbench immediately submits all nodes that are ready for execution, that is, all their prerequisite nodes have been executed already. Most simulation projects are scheduled in an order that is close to a stepwise order. This is a typical node dependency scheme where a node depends on a previous node of the same experiment.

However, in some cases, you might want to change this predefined project execution order. Sentaurus Workbench allows you to do this by setting a concurrency mode for experiments in the Run Project dialog box in the **Concurrency Mode** field:

- **Unlimited**: Default. Sentaurus Workbench uses the standard job-scheduling approach. In general, all project experiments run simultaneously, and you observe a stepwise execution order (see Figure 65 on page 186).
- **Limited**: You specify the maximum number of running experiments that can be executed simultaneously. Sentaurus Workbench ensures this limit is respected during the entire project execution. This number must be between 1 and *N* – 1, where *N* is the total number of experiments. The default is 1, which gives you a strict experiment-wise project execution order (see Figure 66).

Figure 66 Run Project dialog box with Limited selected as the concurrency mode



By default, Sentaurus Workbench uses the Unlimited concurrency mode.

To change the mode:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings**.
3. Set **Default Concurrency of Running Experiments** to **Limited**.
4. Click **Apply**.

Note:

You cannot change the concurrency mode of an already running project. To change it, you must stop the project and rerun it in the required mode.

Although the Unlimited concurrency mode generally leads to a stepwise execution order, it is not a fully strict stepwise order. For example, if your project has nonstandard node dependencies, these dependencies can affect the execution order.

The Limited concurrency mode allows your project to be executed in an experiment-wise manner. However, this can affect the overall project scheduling performance.

In the Unlimited concurrency mode, only node dependencies should be fulfilled, and ready nodes are immediately submitted. In the Limited concurrency mode, Sentaurus Workbench does not execute nodes despite that they are ready to run, so as to respect the given limitation. This approach is much less effective than the Unlimited concurrency mode. Typically, the difference in the overall project execution time between these two modes increases with the number of steps in the Family Table view. The more steps you have, the longer it takes to execute the project.

If a project has nonstandard node dependencies, it can lead to exceeding the maximum number of running experiments in the Limited concurrency mode. For example, assume a node depends on all the nodes in the previous step. Such a dependency requires that all experiments are executed up to the current step. Sentaurus Workbench can prevent a deadlock in such a situation by executing a larger number of experiments, which might temporarily exceed the given limit.

Note:

You cannot use the Limited concurrency mode when run limits are set (see Defining Run Limits on page 197).

Configuring the Execution of Jobs

TCAD Sentaurus simulator jobs can run in parallel using either shared-memory parallelization (SMP) or message passing interface (MPI) parallelization. SMP means that the job process launches multiple parallel threads. MPI parallelization is a case of distributed parallelism: the simulator launches several worker processes that run in parallel on different hosts. Each MPI worker process can be either a serial or multithreaded process.

Sentaurus Workbench supports job execution locally and on different job scheduling systems (cluster or farm) (see Chapter 11 on page 262):

- IBM Platform LSF (LSF)
- Oracle Grid Engine and Univa Grid Engine (SGE)

- TORQUE Resource Manager (TM)
- Runtime Design Automation NetworkComputer (RTDA)

Typically, these remote schedulers provide an infrastructure for executing SMP and MPI parallelization jobs. However, the schedulers require that you allocate the required resources before job execution, such as the number of requested parallel slots (CPUs) and the amount of memory (RAM). Sentaurus Workbench takes over this task and requests appropriate computational resources during the submission of a job.

Note:

To run SMP and MPI jobs on the SGE scheduler, parallel environments must be configured on the SGE side (see Appendix G on page 341).

To configure the job execution:

1. Open the Tool Properties dialog box and click the **Parallelization** tab.
2. The **Activate Parallel Settings** option is selected by default.

   If you clear this option, Sentaurus Workbench works in a legacy mode and assumes that the job is a serial process. This legacy mode is not recommended.
3. In the **Threads** field, specify the number of static threads the tool uses generally.

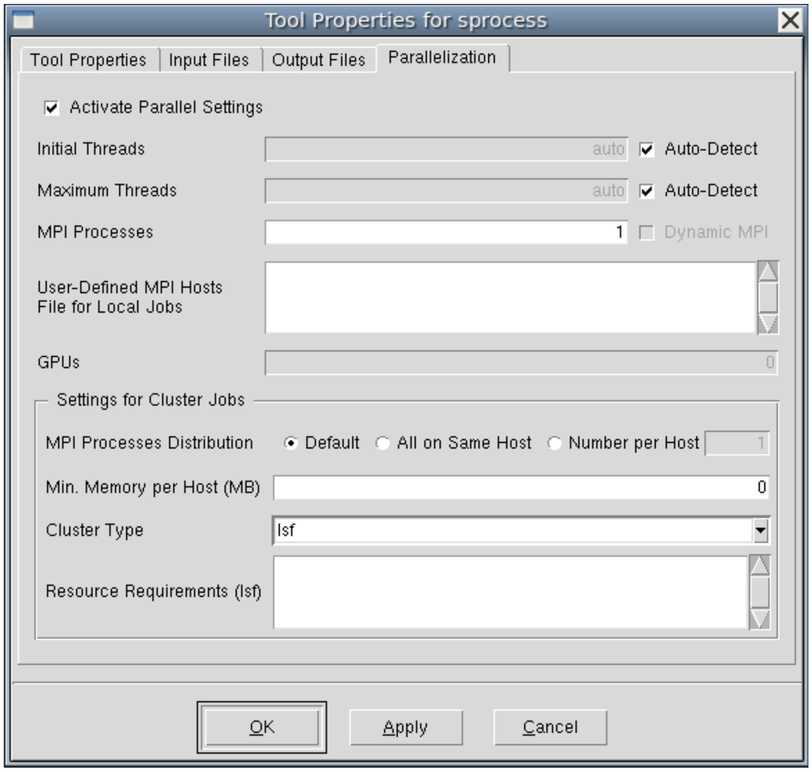
   You cannot change this number during the simulation runtime. However, some tools such as Sentaurus Process, Sentaurus Interconnect, Sentaurus Band Structure, and Sentaurus Device QTX allow you to change the number of threads used during the simulation. For these tools, the **Threads** field appears in the dialog box as the **Initial Threads** field, which specifies the initial number of threads the tool uses generally.

   The **Auto-Detect** option is selected by default. It instructs Sentaurus Workbench to assume the optimal number of threads (see Auto-Detection of Threads for Shared-Memory Parallelization on page 195).

   You can clear this option to specify the number of threads in the **Threads** or **Initial Threads** field manually:

   - 1: A serial process
   - > 1: A shared-memory (multithreaded) process
4. In the **Maximum Threads** field, specify the maximum-allowed number of threads that the tool must not exceed during the simulation.

   The **Auto-Detect** option is selected by default. It instructs Sentaurus Workbench to assume the optimal number of maximum threads (see Auto-Detection of Threads for Shared-Memory Parallelization on page 195). You can clear this option to specify the maximum number of threads in the **Maximum Threads** field manually.



5. In the **MPI Processes** field, specify how many parallel processes the job must launch:
   - 1: (default) A serial or an SMP job
   - > 1: An MPI job, where the **Threads** (or **Initial Threads**) and **Maximum Threads** fields define the number of threads per MPI process
6. Select **Dynamic MPI** if it applies to your simulation task.

   This option is available only for tools that support MPI-2 dynamic process management, such as Raphael™ FX.
7. Sentaurus Workbench offers several options for submitting an MPI job:

   a. If you want to launch all MPI processes on the local host, then submit an MPI job to a local queue. Leave the **User-Defined MPI Hosts File for Local Jobs** field empty.

   b. If you want to launch MPI processes on dedicated hosts, then specify a list of hosts with the number of processes to run on each host in the **User-Defined MPI Hosts File for Local Jobs** field, in the format as described in the *TCAD Parallelization Environment Setup User Guide*. For example:

   ```
   tcad3g1:4
   tcad3g2:2
   tcad3g3:2
   ```

Then submit an MPI job to a local queue.

The MPI process manager `mpiexec.hydra` launches MPI worker processes on remote hosts using the SSH launcher. It is required that you have a silent SSH login without a password to the given hosts. To configure a silent SSH login, see Appendix F on page 339.

By default, Sentaurus Workbench checks for a silent SSH login to all hosts you provided before launching the MPI job. If this check is not required, then you can switch it off in preferences, by expanding **Scheduler** > **Local Jobs** > **Check SSH Connectivity for MPI Hosts** = No.

c. You can run MPI jobs on the job schedulers SGE and LSF. You need to submit an MPI job to an SGE or LSF queue, respectively. Sentaurus Workbench composes the MPI hosts file with the hosts allocated by the job scheduler. In this case, Sentaurus Workbench ignores the hosts specified in the **User-Defined MPI Hosts File for Local Job**s field.

   By default, the MPI process manager `mpiexec.hydra` launches MPI worker processes on scheduler hosts using the native launcher (`blaunch` for LSF; `qrsh` for SGE). However, you can change it to SSH in preferences, by expanding **Scheduler** > **LSF Jobs** > **Launcher for MPI Processes** or **Scheduler** > **SGE Jobs** > **Launcher for MPI Processes**. In this case, it is required that you have a silent SSH login without a password to the given hosts. To configure a silent SSH login, see Appendix F on page 339.

   If you want to use SSH as the launcher for MPI processes, by default, Sentaurus Workbench checks for a silent SSH login to all hosts you provided before launching the MPI job. If this check is not required, then you can switch it off in preferences, by expanding **Scheduler** > **LSF Jobs** > **Check SSH Connectivity for MPI Hosts** = No or **Scheduler** > **SGE Jobs** > **Check SSH Connectivity for MPI Hosts** = No.

8. Leave the **GPUs** field, which specifies how many GPUs each process uses. It is deactivated for all TCAD Sentaurus tools. The default is 0, which means no GPU parallelization.

9. For **MPI Processes Distribution**, select how to distribute processes over the available computing resources when you submit a job to a job scheduler:

   - **Default** (default) allows the scheduler to determine how to distribute processes on hosts. The behavior might depend on the scheduler configuration.
   - **All on Same Host** instructs the scheduler to allocate all processes on the same host.
   - **Number per Host** instructs the scheduler as to how many processes to allocate per host.

10. In the **Min. Memory per Host (MB)** field, specify how much memory in megabytes must be reserved per host when the job is submitted to a job scheduler:

    ◦ 0: (default) Sentaurus Workbench does not request memory explicitly. In this case, the submission queue defines the memory policy.

    ◦ > 0: Sentaurus Workbench requests a given amount of memory. Use a decimal notation, for example, 1000 = 1 GB.

11. From the **Cluster Type** list, select a scheduler that is available in your computational environment.

12. To properly launch a job on the selected remote cluster, in the **Resource Requirements** field, enter the appropriate resources in the definition of the queue to which you submit your job.

    Enter a string using the syntax of the target scheduler. The content of this string is added as an additional resource requirement on job submission. Default: Empty for all supported clusters.

    In addition, you can provide specific resources that the tool needs when launching on different job schedulers. The given resources are added to those specified in the queue to which you submit your job. You can specify particular resources for all job schedulers available in your corporate environment (LSF, SGE, RTDA, or TM).

13. Repeat Steps 11–12 for other remote clusters that are available in your computational environment.

14. Click **Apply**.

Note:

The settings made in Steps 9–12 take effect only when you launch the tool on the corresponding remote cluster.

The following tools support static SMP:

- Garand MC
- Sentaurus Device
- Sentaurus Device Electromagnetic Wave Solver (EMW)
- Sentaurus Mesh
- Sentaurus Topography 3D

The following tools support dynamic SMP:

- Sentaurus Band Structure
- Sentaurus Device

- Sentaurus Device QTX (Subband-BTE solver)
- Sentaurus Interconnect
- Sentaurus Process
- Sentaurus Visual

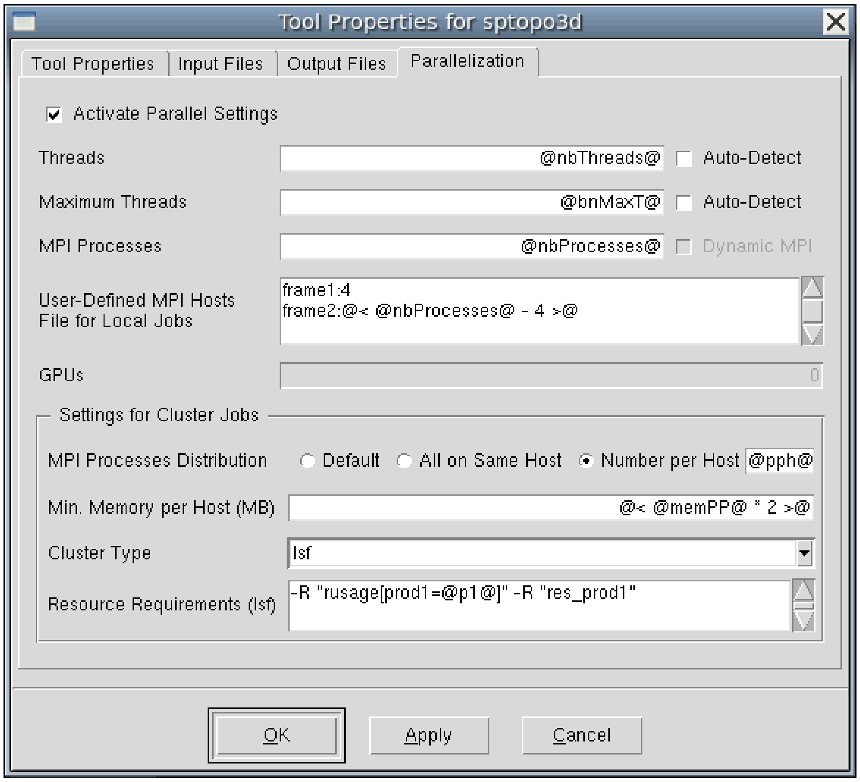
The following tools support MPI parallelization:

- Sentaurus Device
- Sentaurus Device Electromagnetic Wave Solver (EMW)
- Sentaurus Device QTX (NEGF solver)
- Sentaurus Interconnect
- Sentaurus Process (only Monte Carlo implantation)
- Sentaurus Topography 3D

Sentaurus Workbench deactivates settings that do not apply to a given tool.

For greater flexibility, Sentaurus Workbench allows you to parameterize some job execution settings, namely, the number of processes, number of threads per process, memory, and scheduler-specific resource requirements (see Figure 67).

Figure 67 Tool Properties dialog box with parameterized parallelization settings



Auto-Detection of Threads for Shared-Memory Parallelization

Sentaurus Workbench assesses the number of threads required for job execution and deactivates the **Threads** or **Maximum Threads** field, so you cannot explicitly specify the number of threads in these fields.

With the **Auto-Detect** option selected, Sentaurus Workbench performs the following steps for each node you launch on a cluster:

1. Preprocess the node (if needed).
2. Inspect the node command file and identify parallelization-specific keywords that might depend on the tool syntax.
3. Check command-line arguments in the tool properties (some tools support command-line options specifying the number of CPUs for parallelization).
4. Inspect the input TDR files (some tools store parallelization keywords there).
5. Consider the highest of all found numbers as the required number of threads.

Sentaurus Workbench allows auto-detection of CPUs only for tools supporting SMP.

Be careful with thread-related instructions in the case of Sentaurus Process split nodes. Sentaurus Process stores thread-related settings in a TDR file. If you want to change the number of requested CPUs for a job explicitly, ensure that the input TDR file does not contain other thread-specific instructions inherited from previous nodes.

For example, the following Sentaurus Process code executes in four nodes with assessed multithreading requirements as 4, 8, 8, and 8 corresponding to the `numThreadsMC` value in the third node:

```
...
math numThreads=4
...
#split @implant@
...
math numThreads=1
math numThreadsMC=8
...
#split @mask@
...
math numThreads=1
math numThreadsMC=1
...
#split @temp@
...
math numThreads=2
math numThreadsMC=2
...
```

If you want the last two nodes to execute with one and two threads, you must change the `numThreadsMC` value in the last two splits:

```
...
#split @mask@
...
math numThreadsMC=1
...
#split @temp@
...
math numThreads=2
...
```

Note:

For complex process flows, it is recommended to split the flow into simpler parts with definitive multithreading requirements. Otherwise, Sentaurus Workbench might provide an unexpected assessment of the resource requirements.

Limiting the Number of Threads Requested

The auto-detection of threads might lead to requesting too many threads to be allocated, which can potentially exceed the limit provided by the hardware.

To prevent such a situation, you can specify the maximum number of threads to be allocated in preferences:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings** > **CPUs Auto-Detection Upper Limit**.
3. Set the limit.

   The default is 0, which means the detected number of threads is not limited. Otherwise, the auto-detection algorithm is restricted to the value specified.
4. Click **Apply**.

Defining Run Limits

Sentaurus Workbench helps you to organize effective use of both available TCAD Sentaurus licenses and corporate computational resources for a group of users sharing the same installation of TCAD Sentaurus. This goal is achieved by controlling the maximum number of simultaneously running simulations of a certain tool by a user. Hereafter, this maximum number is referred to as *run limits*.

Note:

You cannot use run limits together with the Limited concurrency mode (see Concurrency Mode for Experiments on page 187).

Defining the User Quota

The following example illustrates how run limits are taken into account when running simulations. Assume you want to run no more than four Sentaurus Process instances simultaneously. When the next Sentaurus Process job is ready to be submitted, Sentaurus Workbench checks the number of your currently running Sentaurus Process jobs. When this number is less than four, Sentaurus Workbench submits this job. Otherwise, the job is not submitted and remains in a *ready* status until the next free job slot to run the simulation.

Sentaurus Workbench allows a flexible definition of run limits (see Run Limits Settings on page 245). The simplest case is defining constant run limits for certain tools. You can go further and define a run limits timetable where run limits depend on the day of the week, the time of the day, holidays, and so on. With the timetable approach, TCAD users can establish a flexible and dynamic run limits schedule taking into account resource availability.

Depending on corporate resource-sharing strategies, you can define run limits on a centralized level (global or site) or allow users to apply their own run limits on the user level. As a TCAD installation administrator, you have full control over run limits settings. When needed, you can prohibit redefinition of global run limits on a user level. You can modify run limits at any time without interrupting already running projects. Run limits restrictions that are agreed upon and applied inside a group provide a fair distribution of available licenses between users, as well as the possibility of the temporary concentration of available resources for urgent work.

Sentaurus Workbench calculates the number of currently running simulations of a certain tool on a user-based mode. All the simulations that you launch in Sentaurus Workbench are taken into account. The number of currently running jobs is controlled by the daemon process `swblm`, which runs in the background. Before submitting a simulation job, Sentaurus Workbench checks with `swblm` as to whether the number of currently running jobs of the same type exceeds the allowed maximum number and, if not, Sentaurus Workbench submits the job.

When all available job slots are occupied, Sentaurus Workbench tries to resubmit a job in case an available slot appears. You can set up a maximum number of times that Sentaurus Workbench tries to resubmit a job. After that number is reached, Sentaurus Workbench can either report the job as failed (default behavior) or proceed with the job submission ignoring the run restrictions.

Note:

Run limits are defined as the maximum number of slots for a given tool to run simultaneously. It is not related to the number of specific licenses the tool checks out at runtime.

Run limits are switched off by default, which means that no restrictions are applied. To activate run limits, they must be specified in the global, site, or user run limits files (see Run Limits Settings on page 245).

Note:

Only those simulations you launch *from* Sentaurus Workbench are taken into account by run limits control in Sentaurus Workbench. If you launch a tool outside of Sentaurus Workbench, for example, on the command line, Sentaurus Workbench is not aware of this job.

Run limits defined on a centralized level (global or site) or on user levels specify a *user quota*. Sentaurus Workbench ensures that you do not exceed your user quota when running your simulation projects.

Defining a Submission Delay

Together with the user quota, you can define a delay between two sequential submissions of a given tool. This delay might be helpful if you want to set up a fair distribution of computational resources among TCAD users and to ensure that no single user occupies all the available running slots on the cluster for a short period. Defining a submission delay also reduces the load on overworked clusters.

Sentaurus Workbench allows you to define a submission delay for tools with and without run limits. Unlike run limits, the submission delay is constant per tool and cannot depend on the day of the week, the time of the day, holidays, and so on.

Defining Project Run Limits

Another important aspect of organizing effective use of available TCAD Sentaurus licenses and corporate computational resources is the ability to partition your user quota among different projects that you want to run simultaneously. By default, Sentaurus Workbench does not control how the user quota is distributed among projects. If you launch several projects simultaneously, your user quota might be reached in a single project run, while other running projects must wait until a free slot becomes available.

Sentaurus Workbench allows you to control how your run limits quota is partitioned among your projects by setting up optional *project run limits*.

Project run limits are additional restrictions that affect only a given project and can be defined only if the user quota has been configured (see Defining the User Quota on page 197).

When launching a project, you specify project run limits directly in the Run Project dialog box (see Figure 68 on page 200). If you do not see the project run limits part of the dialog box, click **Show Project Run Limits**.

Note:

Only tools with the defined user quota are available for setting the project run limits in the Run Project dialog box.

By default, the project run limits are switched off. All tools in a project have the default project run limits of 0, which means that no restrictions on the project level apply. In this case, the number of simultaneously running tools is defined exclusively by the user quota.

If you want to set up a project run limits for specific tools, then enter the appropriate limits in the corresponding entries, using 0 for other tools. In general, specify project run limits as follows:

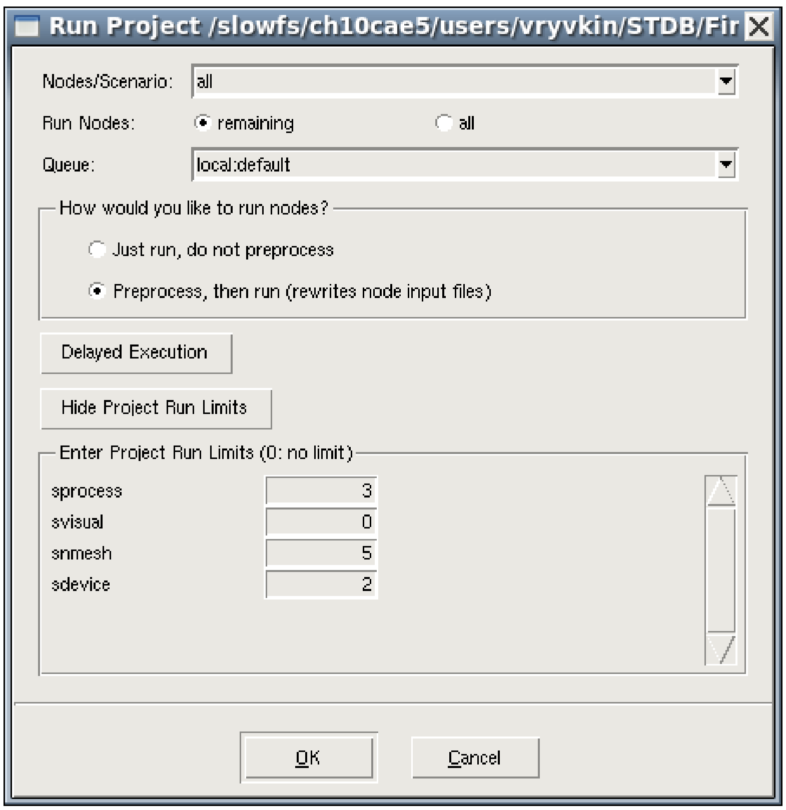
- **0**: You do not want to limit this tool in the current project.

- **Not 0**: You want to limit the number of concurrently running instances of this tool in the current project to the given number. For low-priority projects, you would specify a smaller number than for high-priority projects. The minimum value is 1. The maximum number is set by the corresponding user quota defined by the run limits on the global, site, or user level.

Note:

If you specify project run limits that exceed the corresponding user quota, the user quota takes effect. Regardless of what you specify for project run limits, Sentaurus Workbench ensures that the user quota is not exceeded.

Figure 68 Run Project dialog box with options for project run limits shown



The project run limits are stored in the Sentaurus Workbench project and are applied to this specific project only. The next time you launch the project, you will see a dialog box that requires confirmation as to whether you want to use the same project run limits as you used on the previous run. You can change or remove the project run limits by specifying 0 for all tools.

The project run limits take effect during the overall project execution. However, you can change the project run limits of a running project in one of the following ways:

- If you plan to submit additional project nodes for execution:

  Select those nodes and click **Run**. Change the project run limits in the Run Project dialog box and click **OK** to launch the selected nodes.

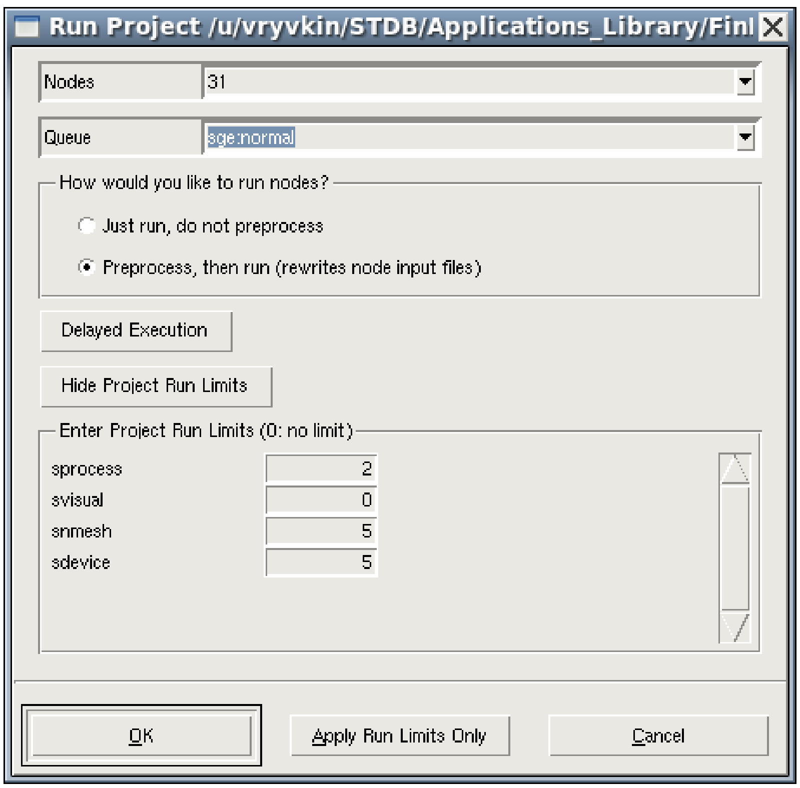
  Sentaurus Workbench launches the selected nodes and applies the new project run limits.

- If you do not want to submit additional project nodes for execution:

  Select any node and click **Run**. Change the project run limits in the Run Project dialog box and click **Apply Run Limits Only**.

  Sentaurus Workbench sends the new project run limits to the running project (see Figure 69).

Figure 69 Changing the run limits of a running project in the Run Project dialog box



Note:

It might take some time until the new project run limits take effect for a running project, in case you decided to lower the previously specified project run limits. Sentaurus Workbench does not interrupt already running jobs. New project run limits are applied as soon as possible, that is, when some of the running jobs are finished.

Applying Run Limits to Viewers

By default, Sentaurus Workbench applies run limits when running simulation projects. Your quota is not applied to viewers that you launch to visualize node files. For example, if your Sentaurus Visual quota is 10, then you can concurrently execute up to 10 Sentaurus Visual nodes and launch an unlimited number of Sentaurus Visual viewers to visualize node structures.

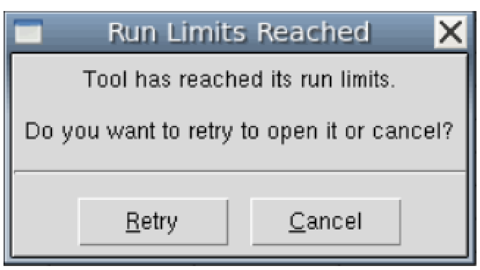
You can configure Sentaurus Workbench to apply run limits to both batch and interactive sessions of tools (see Run Limits Settings on page 245). Then, Sentaurus Workbench run limits take effect for all interactive viewer sessions that you launch from within Sentaurus Workbench, including from the **Nodes** > **Visualize** and **Extensions** menus.

If your tool quota has already been reached, then Sentaurus Workbench displays a warning dialog box when you try to launch an interactive session of this tool (see Figure 70).

Note:

Run limits do not apply to interactive tool sessions that you launch outside of Sentaurus Workbench, for example, from the command line.

Figure 70 Run Limits Reached dialog box



Changing the Order of the Execution of Nodes

Note:

This feature is available only if run limits are active. When run limits are not active, use the concurrency mode to access the same functionality (see Concurrency Mode for Experiments on page 187).

When there are no node dependencies or there is a choice as to which nodes to run first, you can specify how Sentaurus Workbench should submit nodes. You can choose from the following node orders:

- **Depth_First**: Sentaurus Workbench tries to complete the current experiment and then proceeds to the next experiment.
- **Breadth_First**: Sentaurus Workbench tries to execute as many experiments as possible. This is the default.

To change the order of submitted nodes:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings**.
3. Set **Nodes Running Order Under Run Limits** to **Depth_First**.
4. Click **Apply**.

When a project has node dependencies (which is the case for most Sentaurus Workbench simulation projects), you will not see a difference between the different node orders because node dependencies take priority and define the node submission order.

Defining Runtime Limits

Some simulations run longer than others. Distinct simulations run even longer. The reason might be because of the complexity of the task or the solver does not converge. Sentaurus Workbench allows you to detect long-running simulations in a timely manner, so that you can decide whether to stop or continue running such jobs. You can do this by controlling the simulation runtime limit. Hereafter, this limit is referred to as *runtime limits*.

The following example illustrates how runtime limits are taken into account when running simulations. Assume you want to be warned when Sentaurus Device tool instances run longer than 2 days. Sentaurus Workbench checks for how long your Sentaurus Device jobs are running and informs you as soon as, at least, one of them exceeds the predefined runtime limits.

You can configure how Sentaurus Workbench informs you (by sending an email or printing a warning to the execution log file) and the periodicity of the notification. When you receive a notification, check the running job and decide whether it should continue. You can also instruct Sentaurus Workbench to terminate a long-running job after a predefined number of notifications.

By default, Sentaurus Workbench does not detect long-running jobs. You can define runtime limits in the run limits file (see Run Limits Settings on page 245).

Note:

Runtime limits are supported starting from the runtime limits file version 1.0. To activate runtime limits, ensure that the `Version` attribute of the `RunLimitsTable` tag has a value of 1.0 or higher.

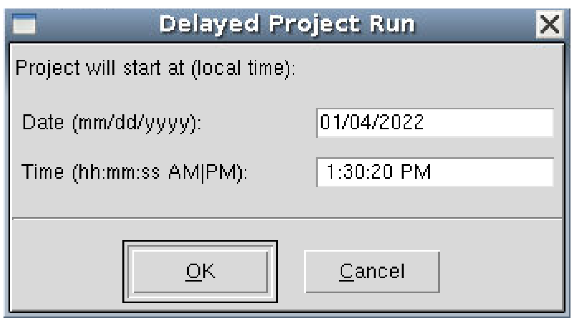
You can define both overall runtime limits for all tools and tool-specific runtime limits.

Unlike run limits, runtime limits are constant per tool and do not depend on the day of the week, the time of day, holidays, and so on.

Delaying the Execution of Projects and Nodes

The **Delayed Execution** button on the Run Project dialog box allows you to define the time at which a project or selected nodes are submitted (see Figure 71).

Figure 71 Delayed Project Run dialog box



Sentaurus Workbench preprocesses launched nodes immediately, while nodes submission is suspended until the given time. By default, the Delayed Project Run dialog box populates the fields with the current date and time, that is, nodes are submitted immediately.

Delayed execution in Sentaurus Workbench has some limitations as follows:

- You cannot submit multiple node groups with a different submission time.
- Sentaurus Workbench does not mark nodes waiting for submission with a specific status (or color), so it might be difficult to distinguish such nodes without analyzing of the project submission log file (`glog.txt`).

Given that the main goal of delayed execution is to optimize the use of available computational resources and TCAD Sentaurus licenses, it is highly recommended to use a dynamic run limits schedule to maximize flexibility (see Defining Run Limits on page 197).

Configuring a Delay Between Simulations

You might have a situation where node execution fails unexpectedly because the simulation process cannot source some input files. For example, the TDR file stored on the previous node cannot be loaded for the next node. Such a problem could appear on slow (or busy) network file systems, which require more time to synchronize file content between file servers.

To configure a delay between simulations:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings**.
3. Set **Delay After Simulation (msec)** to the time to wait after completing the simulation.
4. Click **Apply**.

   The longer the delay, the less likely a node failure will occur.

Note:

A correctly configured network file system does not require such fine-tuning.

Protecting Executed Nodes

When working with a large project, you might accidentally select and submit nodes that have been completed already, which will delete the previous results. Such an error might cost substantial time and computational resources.

Sentaurus Workbench provides a mechanism to protect executed nodes with the *done* status from being rerun. By default, this mechanism is switched off.

To protect executed nodes:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings** > **Protecting of Done Nodes**.
3. Set **Allow Rerun Done Nodes** to **No**.
4. Click **Apply**.

With this protection, Sentaurus Workbench does not launch completed nodes with the *done* status.

At the bottom of the project log file (`glog.txt`), Sentaurus Workbench prints a summary that includes nodes that did not start because they had the *done* status:

```
SCHEDULING REPORT
+++ done: 10 11 13 22 23 25
+++ protected: 12 24
```

The only way to rerun completed nodes is to clean them up explicitly (see Cleaning Up the Output of Nodes on page 217).

You can set up a notification to inform you when you try to submit *done* nodes before the execution starts.

To switch on this notification:

1. Choose **Edit** > **Preferences** or press the F12 key.
2. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings** > **Protecting of Done Nodes**.
3. Set **Show Warning if No Rerun Allowed** to **Yes**.
4. Click **Apply**.

A warning message appears only if protection for executed nodes is switched on.

Protecting executed nodes does not prevent these nodes from being preprocessed. In general, it might lead to inconsistency between the preprocessed node command file and the actual simulation results.

Preprocessing Projects

Preprocessing is the initial step before jobs actually run and can be performed separately from execution either from the user interface (choose **Project** > **Operations** > **Preprocess** or press Ctrl+P) or from the command line (call the Sentaurus Workbench preprocessor (`spp` utility)).

Note:

The `spp` utility is executed as a separate process from the user interface, and the result of `spp` is stored in the file `preprocessor.log`, which can be accessed by choosing **Project** > **Logs** > **Preprocessor**.

You can preprocess multiple selected projects from the projects browser.

The `spp` utility preprocesses the given tool input file to standard output or performs a global preprocessing pass in a project. For example:

```
spp "BOR_IMPL_ENRGY80" "N_DOSE1e15" @STDB@/nmos,process/lig.cmd
spp -verbose @STDB@/nmos,process
spp -expr "scnr_name" @STDB@/nmos,process
```

Table 10 Command-line options of spp

Option	Description
`-e[xpr]`	Preprocess based on an expression
`-h[elp]`	Displays help information
`-i[nput]`	Generates all of the initial, preprocessed `pp*` files in project
`-n[odes]`	Preprocess just these nodes
`-onlytcl`	Preprocess only Tcl blocks in command files
`-p[ack]`	Does not generate empty lines while preprocessing
`-v[ersion]`	Displays version information
`-verbose`	Displays additional initialization and runtime information
Arguments	
`FILENAME`	Marked tool input file
`PROJECT`	Project directory

Terminating Projects and Nodes

You can terminate projects and nodes as required.

Terminating Projects

You can terminate a running project from the Project Editor or the projects browser.

To terminate a project from the Project Editor:

1. Select a project.
2. Double-click the selection, or right-click and choose **Open**.

The project opens in running mode.

3. Choose **Project** > **Operations** > **Abort**, or press Ctrl+T, or click the corresponding toolbar button.
4. Confirm that the project is to be terminated.

To terminate a project from the projects browser:

1. Select a project.
2. Choose **Project** > **Operations** > **Abort**, press Ctrl+T, or right-click and choose **Project** > **Abort**.
3. Confirm that the project is to be terminated.

Note:

You can terminate multiple selected projects from the projects browser.

Terminating Nodes

You can terminate specific running nodes in a project from the Project Editor, the Scheduler, or the command line.

To terminate a running node from the Project Editor:

1. Select a project.
2. Double-click the selection, or right-click and choose **Open**.

   The project opens in running mode.
3. Select a node.
4. Choose **Nodes** > **Abort**, press Ctrl+T, or right-click and choose **Abort**, or click the corresponding toolbar button.
5. Confirm that the node is to be terminated.

To terminate a running node from the Scheduler:

1. Select a node from the right pane of the Scheduler, which displays the current list of running nodes.
2. Choose **Nodes** > **Abort**, press Ctrl+T, or right-click and choose **Abort**.
3. Confirm that the node is to be terminated.

Note:

If you want to terminate the entire project, it is more efficient to implement that at the project level (see Terminating Projects on page 207) rather than selecting all the project nodes and terminating them.

Unexpected Termination of Running Projects

If the terminal from which Sentaurus Workbench was launched is closed or crashes, it might lead to the unexpected termination of Sentaurus Workbench and the running projects you launched in this Sentaurus Workbench instance. In this case, `gsub` processes receive the signal to terminate from the operating system.

If closing the terminal is inevitable and you must terminate running projects, you can force Sentaurus Workbench and `gsub` to run in a *daemon* mode by setting the environment variable `SWB_DAEMONIZE` to 1:

```
setenv SWB_DAEMONIZE 1
```

This will launch Sentaurus Workbench and the `gsub` processes that are disconnected from the parent terminal process from where you launched Sentaurus Workbench. As a result, your running projects will be completed even if the terminal process crashes or is terminated.

Updating Node Statuses and Extracted Variables

Sentaurus Workbench updates the node statuses of running projects and extracts simulation results automatically as soon as the node status changes.

Simulation results are extracted from a node output file as soon as the simulation is completed. However, for long-running simulations, it might be helpful to see completed results before the simulation itself is finished. You can force the status updating of all project nodes as well as the values of extracted variables at any time. In this case, Sentaurus Workbench reads the statuses of nodes and the values of extracted variables from the project files.

To update the status of nodes at any time:

► Choose **View** > **Refresh** or press the F5 key.

Note:

You should refresh project nodes rather than reload the project when you need to monitor the progress of running projects. Refreshing the status of nodes is much faster than reloading a project.

Customizing the Execution of Projects and Nodes

You can customize the execution of projects and nodes.

Customizing Project Execution

Sentaurus Workbench allows you to customize project execution by defining optional prologue and epilogue Tcl scripts in the tool database. The syntax is:

```
set WB_tool(gsub,prologue) { ... arbitrary Tcl script ... }

set WB_tool(gsub,epilogue) { ... arbitrary Tcl script ... }
```

The prologue Tcl script is evaluated at the beginning of the execution session when you launch the project (start of `gsub` process). The epilogue Tcl script is evaluated at the end of the execution session (end of `gsub` process).

The prologue script can be useful to prepare for project execution, for example, to clean up the disk space, while the epilogue script can be used for user-specific report generation on an executed project.

Customizing Node Execution

A similar approach to project execution can be used to customize node execution. Sentaurus Workbench allows you to customize node execution by defining tool-specific prologue and epilogue Tcl scripts in the tool database. The syntax is:

```
set WB_tool(<tool>,prologue) { ... arbitrary Tcl script ... }

set WB_tool(<tool>,epilogue) { ... arbitrary Tcl script ... }
```

For example, the prologue and epilogue Tcl scripts defined in the tool database for Sentaurus Mesh (`snmesh`) are:

```
set WB_tool(snmesh,prologue) { snmesh_prologue @node@ @commands@
@tdr@ @tdrboundary@ }

set WB_tool(snmesh,epilogue) { extract_vars "$nodedir" @stdout@
@node@ }
```

Both prologue and epilogue scripts are optional.

If you define a tool prologue Tcl script, Sentaurus Workbench evaluates it immediately before the node simulation task starts.

You can define an epilogue script for almost all supported tools. If you define a tool epilogue Tcl script, Sentaurus Workbench evaluates it immediately after the node simulation task finishes only if the node is successful (node status is *done*). However, you can instruct Sentaurus Workbench to evaluate a tool epilogue Tcl script for failed and terminated simulation tasks as follows:

1. In the SWB Preferences dialog box, expand **Scheduler** > **General Settings**.
2. Set **Run Epilogue on Failed and Aborted Nodes** to **true**.
3. Click **Apply**.

Note:

Be careful when redefining tool prologue and epilogue Tcl scripts in the tool database. The best practice is to append your Tcl commands to the end of the existing Tcl script (if any).

Rerunning Failed Nodes Automatically

If a simulation fails because of convergence issues or other reasons, then sometimes there is the possibility that rerunning the same simulation will succeed. You can configure the automatic rerunning of failed nodes by setting the maximum number of run trials per individual tool in the tool database. By default, Sentaurus Workbench runs every simulation once and does not try to rerun failed simulations.

For example, this line instructs Sentaurus Workbench to launch a Sentaurus Topography 3D simulation up to three times:

```
set WB_tool(sptopo3d,rerun_failed,max_trials) 3
```

If the simulation successfully completes within three runs, then Sentaurus Workbench displays the corresponding node as *done*, otherwise the node has the status *failed*. The node `.job` file specifies the information about the reruns, so you can always find the number of failed simulation executions before the successful one.

It might be helpful to distinguish the reason for the simulation failure and allow Sentaurus Workbench to rerun the simulation only if a certain condition applies. For that, you can set up a Tcl script in the tool database that implements a criterion based on the results of the previous execution. If the criterion matches, then Sentaurus Workbench reruns the simulation; otherwise, it reports the node status as *failed*.

You can do this in the tool database in a similar way as with tool prologue and tool epilogue scripts (see Customizing Node Execution on page 210):

```
set WB_tool(sptopo3d,rerun_failed,criteria) {... arbitrary Tcl script
    ...}
```

The script must return one of the following:

- `1`: Rerun the node
- `0`: Do not rerun the node; report node as *failed*

By default, the rerun criterion script is as follows:

```
set WB_tool(sdevice,rerun_failed,criteria) { return 1 }
```

This means that the node must be reexecuted if the maximum number of trials has not been reached yet.

Viewing Project Files

To view a project log file:

1. Select a project.
2. Choose **Project** > **Logs** > **Project**, press Ctrl+J, or right-click and choose **Project** > **View Log**.

To view a project history file:

1. Select a project.
2. Choose **Project** > **Logs** > **History**, press Ctrl+H, or right-click and choose **Project** > **View History**.

Viewing the Project Summary

The project summary provides a short description of a project that was run. The summary file is stored under `gsummary.txt` in the project directory and is generated automatically when the project finishes.

To view the project summary:

- Choose **Project** > **Properties** > **Summary** or press Ctrl+Y.

The following information is provided in this file:

- Project details:
    - Current status
    - When the last modification occurred

  - Who modified the file
  - On which host it was run
- Total number of nodes: active nodes and virtual nodes
- Nodes by status: list of nodes sorted by status
- Hosts and execution information followed by a list of hosts and runtime on each host and number of nodes executed
- Total runtime

Recognizing Suspended Jobs

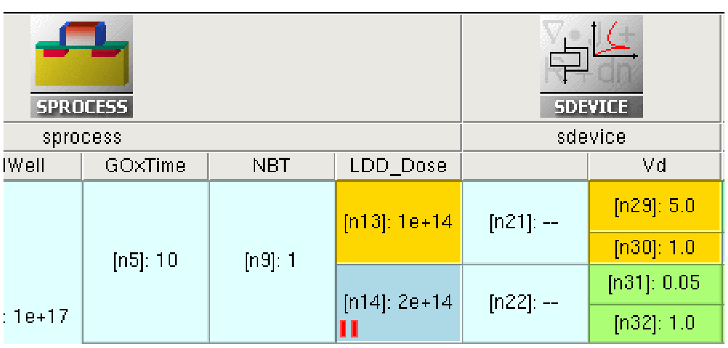
For some TCAD Sentaurus simulators, such as Sentaurus Process and Sentaurus Device, you can suspend and resume their execution. The simulator releases its license during the suspension period and checks it out again when you resume job execution.

Sentaurus Workbench can recognize suspended jobs that were launched from Sentaurus Workbench locally or remotely. Information is communicated to users in the project execution log file (`glog.txt`). For example:

```
Node 50 has been SUSPENDED
```

Suspended nodes are displayed with a red pause icon in the lower-left corner of the node cell (see Figure 72).

Figure 72 Icon indicating that node has been suspended: second cell under LDD_Dose column



As soon as you resume a suspended job, Sentaurus Workbench recognizes it and provides the following output to the project execution log file and removes the icon from the node cell:

```
Node 50 has been RESUMED
```

Note:

Sentaurus Workbench does not allow you to suspend and then to resume running nodes.

Recognizing suspended jobs is an experimental feature with some limitations:

- You cannot suspend and then resume running nodes directly in Sentaurus Workbench. Instead, you must do it manually using the appropriate commands, which might depend on where you run your jobs.
- You must resume suspended jobs before stopping them in Sentaurus Workbench.

To suspend and resume running jobs, enter the correct command on the command line. The following table specifies how to stop and continue a running job with the job identifier 12345. To display job identifiers as node data in Sentaurus Workbench, choose **View** > **Tree Options** > **Job Identifier**.

Where a job runs	To suspend a job	To resume a job
Locally	`kill -TSTP -12345`	`kill -CONT -12345`
LSF	`bstop 12345`	`bresume 12345`
RTDA	`nc suspend 12345`	`nc resume 12345`
SGE	`qmod -sj 12345`	`qmod -usj 12345`
TM	`qsig -s suspend 12345`	`qsig -s resume 12345`

Note:

You might experience a delay until the issued command takes effect and suspends or resumes the job running on a remote scheduler. This delay is configurable on the cluster. If you experience an unreasonable delay, contact your IT.

Some schedulers implement job suspension by sending the signal `STOP`. While a job is suspended, the license is not released. To ensure that the license is released when a job is suspended, the scheduler must send the signal `TSTP`. Contact your IT to request configuring the job suspension method.

8

Cleaning Up Projects

This chapter describes how to clean up projects from the user interface or from the command line.

Cleaning Up Projects

You can clean up multiple selected projects from the projects browser.

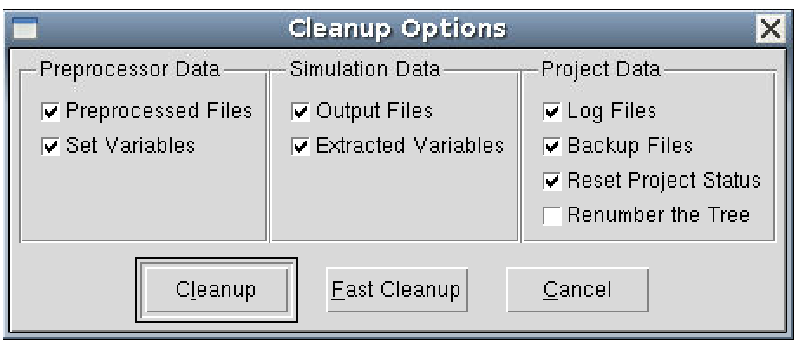
Note:

The options available in the Cleanup Options dialog box differ depending on the organization of a project.

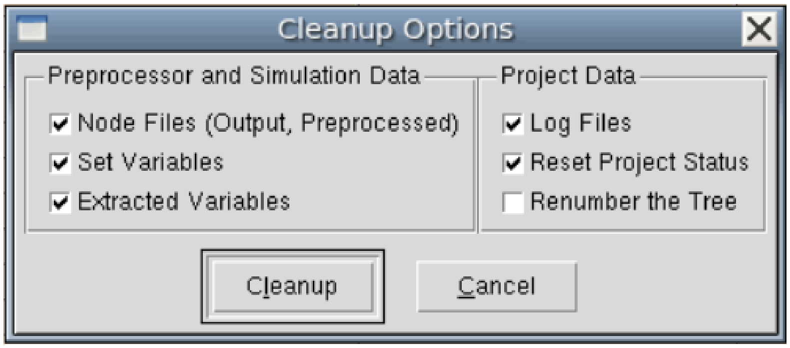
To clean up a project:

1. Choose **Project** > **Operations** > **Clean Up** or press Ctrl+L.

   The Clean Up Options dialog box opens. For a project with traditional organization, the dialog box is as follows:



For a project with hierarchical organization, the dialog box is as follows:



2. Select the required options:

Option	Description
Options for projects with traditional organization	
Preprocessor Data	Deletes preprocessor data:
Preprocessed Files	Deletes all preprocessor files (`pp*`).
Set Variables	Deletes variables set by preprocessor from `gvars.dat`.
Simulation Data	Deletes data remaining after simulation:
Output Files	Deletes all output files (`n<nkey>*`, `pp<nkey>*`, and so on).
Extracted Variables	Deletes extracted variables from `gvars.dat`.
Project Data	Deletes data on the project level (in the project directory):
Log Files	Deletes project log files (`glog.txt`, `gsummary.txt`, `preprocessor.log`, `.history`, and so on).
Backup Files	Deletes the subdirectory `.backup` inside the project directory.
Reset Project Status	Deletes project status file (`.status`).
Renumber the Tree	Renumbers the tree. This option is available only if you select all options under Preprocessor Data and Simulation Data.
Options for projects with hierarchical organization	
Preprocessor and Simulation Data	Deletes preprocessor and simulation data:
Node Files (Output, Preprocessed)	Deletes all preprocessor and output files (`n<nkey>*`, `pp<nkey>*`, and so on).
Set Variables	Deletes variables set by preprocessor from `gvars.dat`.
Extracted Variables	Deletes extracted variables from `gvars.dat`.

Option	Description
Project Data	Deletes reproducible project data:
Log Files	Deletes project log files (`glog.txt`, `gsummary.txt`, `preprocessor.log`, `.history`, and so on).
Reset Project Status	Deletes project status file (`.status`).
Renumber the Tree	Renumbers the tree.

3. Click **Cleanup** (or click **Fast Cleanup** for traditional projects).

   Note:

   The **Fast Cleanup** button is available for traditional projects only. It initiates a faster project cleanup, which is especially noticeable on large projects with many files. The original project directory is removed and then is re-created during this type of cleanup. Therefore, it is highly recommended to close all other applications that are working with project files to avoid unexpected errors or possible loss of data.

Cleaning Up the Output of Nodes

To delete the output and preprocessor data of selected nodes:

1. Select nodes (see Selecting Nodes With Mouse and Keyboard Operations on page 67).
2. Choose **Nodes** > **Clean Up Node Output**.
3. Select whether you want to delete simulation-extracted variables from `gvars.dat`.
4. Select whether you want to delete preprocessed variables from `gvars.dat`.
5. Select output and preprocessor files to be deleted.

   All output and preprocessor files associated with the selected nodes are preset.
6. Click **OK**.

Cleaning Up Projects From the Command Line

The `gcleanup` utility can clean up a project from the command line. For example:

```
gcleanup -default /folder/project
```

Table 11 Command-line options of gcleanup

Option	Description
`-back`	Clean up backup files
`-d[efault]`	Switches on the following options: `-back`, `-log`, `-pp`, `-res`, `-sv`, `-unlock`, `-xv`
`-f[ast]`	Perform fast cleanup **Caution:** The original project directory is removed and then is recreated. You should close all other applications that are working with project files to avoid unexpected errors or possible loss of data.
`-h[elp]`	Display help information
`-log`	Clean up log files
`-n "node list"`	Clean up given nodes only
`-pp`	Clean up preprocessed files
`-ren`	Renumber; automatically switches on the following options: `-pp`, `-res`, `-sv`, `-xv`
`-res`	Clean up output files
`-sv`	Clean up set variables
`-unlock`	Delete the project status file
`-v[ersion]`	Display version information
`-verbose`	Display additional information
`-xv`	Clean up extracted variables
Arguments	
`PROJECT`	Project directory

Detecting Files to Remove

The default Sentaurus Workbench cleanup scenario detects a reproducible part of a project and removes it. All files that appear as the result of project preprocessing and execution are considered to be reproducible files.

To detect files to remove from a project and at node levels, Sentaurus Workbench uses cleanup patterns declared in the tool database. Reproducible node files are set up for each tool. For example, the following line sets up output file patterns for Sentaurus Device nodes:

```
set WB_tool(sdevice,output,files) "n@node@_* pp@node@_* *_n@node@_*"
```

Node files are removed during the overall project cleanup and the cleanup of selected nodes.

Specifying Project Exclude File Patterns

Sentaurus Workbench allows you to specify several exclude file patterns. These patterns define the project files that must be excluded from the project packaging and must be removed during project cleanup. The default Sentaurus Workbench patterns are:

```
set WB_tool(garbage_exclude_patterns) ".nfs* core core.*"

set WB_tool(export_exclude_patterns)  ""

set WB_tool(cleanup_exclude_patterns) ""
```

The `WB_tool(garbage_exclude_patterns)` setting defines the files to appear as the result of system crashes or the deleting of open files. These files must be removed during project cleanup and must not appear in exported project packages.

The `WB_tool(export_exclude_patterns)` setting defines the files not to be included in the package when you export a project (see Exporting and Importing Projects on page 46).

The `WB_tool(cleanup_exclude_patterns)` setting defines the files that must be removed during project cleanup; however, they can be included in exported project packages.

Table 12 summarizes the impact of the project exclude file patterns and the node file patterns on project export and project cleanup operations.

Table 12 Pattern settings affecting project operations

Pattern setting	Cleanup	Export	Export as clean
`WB_tool(garbage_exclude_patterns)`	+	+	+
`WB_tool(export_exclude_patterns)`	–	+	+

Table 12 Pattern settings affecting project operations (Continued)

Pattern setting	Cleanup	Export	Export as clean
`WB_tool(cleanup_exclude_patterns)`	+	–	+
`WB_tool(<tool>,output,files)`	+	–	+

9

Configuring Sentaurus Workbench

This chapter discusses how to configure Sentaurus Workbench.

Preferences

Sentaurus Workbench preferences combine different settings that control the appearance and the overall behavior of Sentaurus Workbench (see Available Preferences on page 225).

Preference Levels

Typically, preferences are configured on the user level. You can configure Sentaurus Workbench as required. However, in some cases, the systems administrator must configure some settings on the global level and propagate them to all users of TCAD Sentaurus. This is especially important for settings related to computational resource sharing, job scheduling, and the location of standard executables. Therefore, Sentaurus Workbench preferences can be defined on three levels: global, site, and user.

Global preferences are usually set up by the systems administrator and are not writable. When your company has multiple groups of users of TCAD Sentaurus, distributed to different sites, it might be useful to customize global preferences for all users at a specific site. To do this, site preferences must be created. The global and site preferences are optional.

To create or access the site tool database, set up the following environment variable to refer to the directory that stores the site settings:

```
% setenv SWB_SITE_SETTINGS_DIR <path_to_site_directory>
```

Like global preferences, site preferences are usually not writable.

Sentaurus Workbench stores preference files at:

- Global level: `$STROOT/tcad/$STRELEASE/lib/glib2/gpref2.$STRELEASE.xml`
- Site level: `$SWB_SITE_SETTINGS_DIR/gpref2.$STRELEASE.xml`

- User level: $STDB/gpref2_<username>.$STRELEASE.xml

Preferences are release specific. Sentaurus Workbench provides automatic conversion of the preferences from the existing preference file of the latest available TCAD Sentaurus release. This conversion is applied separately to the global, site, and user levels. Conversion does not trigger saving the preference file automatically. To store the preference file, in the SWB Preferences dialog box, click **Save** or **Apply**.

User preferences override global and site preferences. Site preferences override global preferences. The hierarchical order is (in descending order):

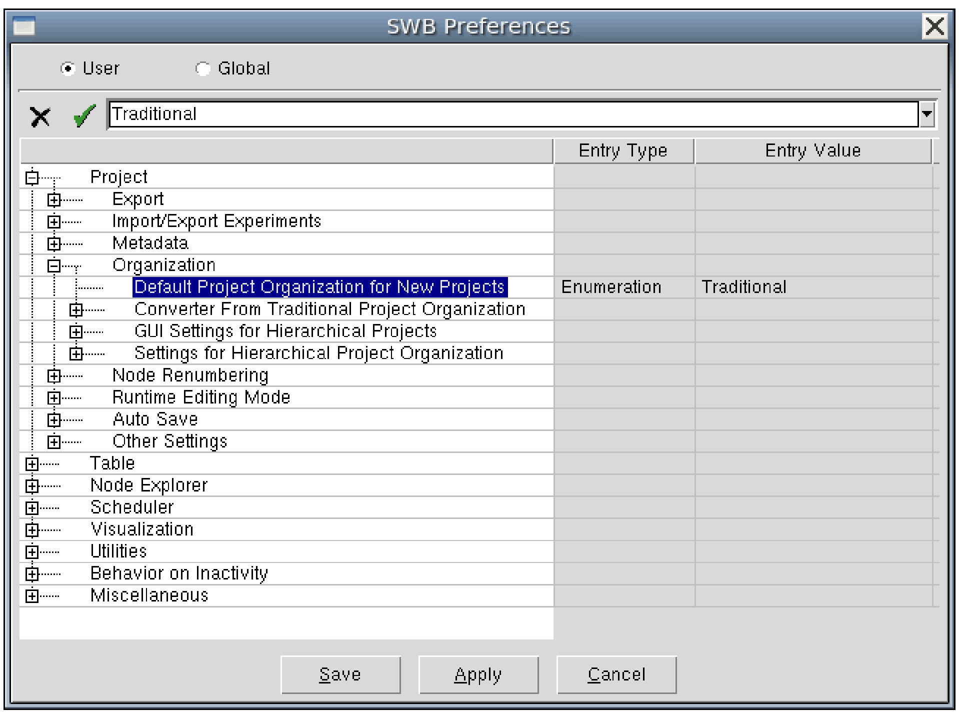
- User preferences
- Site preferences
- Global preferences

Configuring User Preferences

To configure user preferences:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.



2. Select a preference.

3. Enter a new value in the text box.
4. Click **Accept Changes** or press the Enter key.
5. Click **Apply** or Save.

Configuring Global and Site Preferences

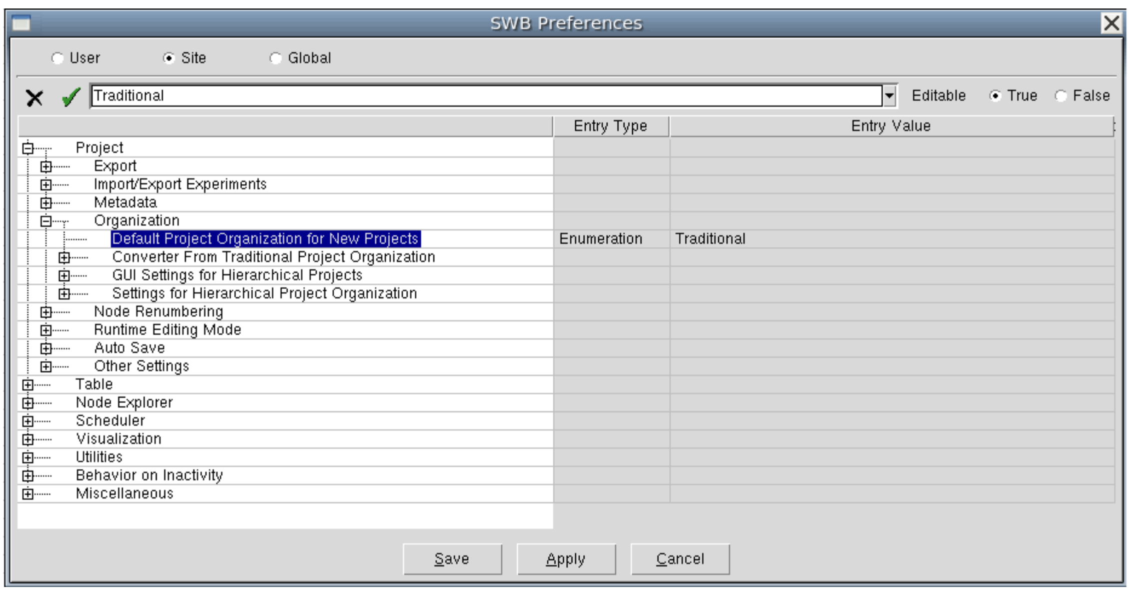
Note:

You can configuring global and site preferences only if you have write permission to the corresponding preference file.

To configure global or site preferences:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.



2. Select **Site** or **Global** at the top of the dialog box.
3. Select a preference.
4. Enter a new value in the text box.
5. (Optional) Select the **False** option for **Editable** in the upper-right corner of the dialog box to prohibit changing the new value on lower preference levels.
6. Click **Accept Changes** or press the Enter key.
7. Click **Apply** or **Save**.

Note:

Click **Apply** when switching between user, site, and global preferences to store your updates in the corresponding preference file.

XML preference files have a distinct main tag depending on the preference level: `<GlobalPreferences>`, `<SitePreferences>`, and `<UserPreferences>`. If you want to use an existing preference file as a template for another level, then you must change the main tag accordingly.

Forcing Global Preferences to All Users

The standard hierarchical order of overriding settings can be changed in the site or global preferences. For each individual preference setting, you can prohibit changing lower precedence levels. To do this, set the attribute **Editable** of the required preference setting to **False**. This setting takes effect regardless of the corresponding settings at lower levels.

As the TCAD administrator, you might want to propagate certain Sentaurus Workbench settings to all users and to ensure that users do not change them at the user level. Use this approach to propagate global preferences at the user level (see Configuring Global and Site Preferences on page 223, upper-right corner of dialog box).

Propagating Default Preferences to Users

As the TCAD administrator, you might want to establish a corporate set of Sentaurus Workbench settings, so that new users start working with Sentaurus Workbench using your default settings rather than the Synopsys defaults. To do this, ensure that your new default settings are specified in the global or site preferences. The standard hierarchical order of overriding settings applies.

Note:

The changed defaults are propagated only to new users of Sentaurus Workbench. If a user already has user preferences stored for the current version or older versions of Sentaurus Workbench, then these stored settings take precedence. In this case, only those settings that are new in the current version are propagated to a user.

Restoring Default Preferences

To restore the default user settings, run Sentaurus Workbench with the `-default` command-line option.

Alternatively, you can remove the user preference file and run Sentaurus Workbench.

Note:

It is highly recommended to store the restored default preferences to ensure that they take effect when using utilities such as `gjob` and `gsub`.

Available Preferences

The following tables describe all of the supported Sentaurus Workbench preferences.

Table 13 Project preferences

Preference	Type	Default	Description
Export			Settings for exporting projects.
Compress Package	Boolean	true	Switches on and off package compression (`gzip`).
Encrypt Package	Boolean	true	Switches on and off package encryption (openssl).
Encryption Key	String	swb	Specifies default encryption key.
Export Project As Clean	Boolean	false	Switches on and off exporting projects as clean projects.
Export Text File:			Settings for export data to a text file.
Include Column Names	Boolean	true	Switches on and off including column names when exporting a DoE table to a text file.
Default Separator	String	,	Selects the default separator when exporting a DoE table to a text file.
Organization			Settings related to hierarchical project organization.
Default Project Organization for New Projects	Enumeration	Traditional	Specifies the default project organization for new projects created by pressing Ctrl+N. It also defines the startup Sentaurus Workbench mode. Options are: • **Traditional** • **Hierarchical**

Table 13 Project preferences (Continued)

Preference	Type	Default	Description
Converter From Traditional Project Organization:			Default settings for dialog box when converting a project from traditional project organization to hierarchical project organization.
Convert Command Files	Enumeration	Yes	Specifies whether the tool input command files should be converted.
Display Changes in Files	Enumeration	Yes	Specifies whether to display applied changes versus original file in the default *diff* application.
Display Conversion Log	Enumeration	Yes	Specifies whether to display a log file with the conversion results.
Existing Result Files	Enumeration	Relocate	Specifies what to do with existing node files. Options are: • **Relocate**: Moves existing files from the root project directory to the nodes hierarchy. • **Remove**: Deletes existing node files.
Save Backup Project	Enumeration	Yes	Specifies whether to save a backup copy of the traditional project organization before conversion.
Check Input File Existence	Enumeration	No	Specifies whether the converter adds the `@pwd@` prefix to non-existing files when converting tool input command files.
GUI Settings for Hierarchical Projects:			Settings for the main window when Sentaurus Workbench is launched in hierarchical mode.
Background	Enumeration	bisque	Background color of the main window of Sentaurus Workbench.
Foreground	Enumeration	System default (white/ gray)	Foreground color of the main window of Sentaurus Workbench.
Distinct Project Status Icons	Enumeration	Yes	Specifies whether to display hierarchical projects with distinct project status icons in the projects browser.

Table 13 Project preferences (Continued)

Preference	Type	Default	Description
Settings for Hierarchical Project Organization:			Settings that take effect on a project with hierarchical organization.
Project Output Files Location	Directory	`@STDB@`	Specifies where to store output files of the project. If it is set to `@STDB@`, output files are stored together with core project files. Otherwise, activates the separate storage of project files.
Location of Temporary Node Files	Directory		Specifies temporary storage of node files when running the project.
Node Renumbering			Settings related to node renumbering
Update Execution Graph	Enumeration	Yes	Specifies whether to update a project execution graph after node renumbering in a hierarchical project. To reduce the time for node renumbering, you can switch off this update and re-preprocess the project.
Update Tags in TDR Files	Enumeration	Yes	Specifies whether to update Sentaurus Workbench tags in TDR files after node renumbering in a hierarchical project. Sentaurus Workbench tags are displayed in Sentaurus Visual. If you do not visualize Sentaurus Workbench tags, you can reduce the time for node renumbering by switching off this update.

Table 13 Project preferences (Continued)

Preference	Type	Default	Description
Runtime Editing Mode			Settings related to a project configuration.
Settings for Editable Mode:			Options for Editable mode.
How to run nodes by default	Enumeration	Preprocess, then run (rewrites node input files)	Specifies the default action in the Run Project dialog box. Options are: • **Just run, do not preprocess**: If global preprocessing succeeds, then runs the node as is, without runtime preprocessing. User changes made in the node input files (if any) take effect. • **Preprocess, then run (rewrites node input files)**: Initiates global and runtime preprocessing, and then runs the node. User changes made in the node input files (if any) do not take effect.
Default Mode	Enumeration	Editable	Specifies the default runtime editing mode for new projects as well as old projects without the mode attributes. Options are: • **Locked**: Maximum automation and consistency level. Running project is locked for changes. • **Editable**: Running project is open for changes, node-level changes in input files are allowed, and all the internal consistency safeguards are switched off.
Auto Save			Setting for automatic saving of projects.
Project Auto Save Interval (min)	Integer	0	The interval in minutes for periodical automatic saving of the currently opened project. When set to `0`, automatic saving is switched off.
Other Settings			Miscellaneous settings.
Polling Interval for Project Database (msec)	Integer	1000	The interval for periodical querying of the project database (`.database`) when it is locked (busy).

Table 13 Project preferences (Continued)

Preference	Type	Default	Description
Show Project Log	Boolean	true	Switches on or off the automatic display of the Project Log window when a project is launched.

Table 14 Table preferences

Preference	Type	Default	Description
Background	Enumeration	System default (white/ gray) or from `.Xdefaults`	Background color of table.
Double Click Action			
On Nodes	Enumeration	Launch Explorer	Specifies the action when double-clicking nodes. Options are: • **Show Properties**: Displays the Node Information dialog box. • **Launch Explorer**: Displays the Node Explorer. • **Run**: Displays the Run Project dialog box. • **Preprocess**: Preprocesses the current node. • **Edit Cell**: Displays an editor that allows you to modify a value directly in the node cell. • **Visualize Results**: Opens the default visualizer for viewing node output files.
On Parameters	Enumeration	Show Properties	Specifies the action when double-clicking parameters. Options are: • **Show Properties**: Displays the Parameter Properties dialog box. • **Add Parameter Value**: Displays the Add Parameter Values dialog box. • **Remove Value**: Displays the Remove Parameter Value dialog box.

Table 14 Table preferences (Continued)

Preference	Type	Default	Description
On Tools	Enumeration	Show Properties	Specifies the action when double-clicking tools. Options are: • **Show Properties**: Displays the Tool Properties dialog box. • **Edit Input File**: Opens the corresponding editor for tool input command files.
On Variables	Enumeration	Show Properties	Specifies the action when double-clicking variables. Options are: • **Show Properties**: Displays the Variable Properties dialog box where you can change the variable value. • **Specify Format**: Displays the Formatting Variables dialog box.
Edit Value in a Cell	Boolean	false	Switches on value-editing in a node cell.
Foreground	Enumeration	System default (white/gray) or from `.Xdefaults`	Foreground color of table.
Font			Font used in project view and projects browser.
Family	Enumeration	System default or customized font selected by user	Specifies the font family to use.
Size	Integer	–	Specifies the font size.
Slant	Enumeration	roman	Specifies slant of the font. Options are: • **roman**: Font is not italicized. • **italic**: Font is italicized.
Weight	Enumeration	normal	Specifies the weight of the font. Options are: • **normal**: Font is not bold. • **bold**: Font is bold.

Table 14 Table preferences (Continued)

Preference	Type	Default	Description
Mouse Wheel	Integer	1	Number of experiments moved when moving the mouse wheel (if supported).
Node Status Color			
List of node statuses	Color	RGB color code	Colors specified as RGB designations for each node status.
Parameter			
Add Parameter/ Value Defaults:			Specifies defaults for Add Parameter/ Values dialog box.
Format	Enumeration	None	Default format of new values. Options are: • **None** • **Scientific** • **Float** • **Integer**
Maximum Number of Values	Integer	100	Maximum number of values to generate at the same time.
Order	Enumeration	Ascending	Default sorting of new values. Options are: • **Ascending** • **Descending** • **Do not sort**
Scale	Enumeration	User Defined List	Default scale (mode) of new values. Options are: • **Linear** • **Logarithmic** • **Gaussian** • **User Defined List**
Scenario			
Remove:			What to do with experiments when removing scenarios.

Table 14 Table preferences (Continued)

Preference	Type	Default	Description
Action on Experiments	Enumeration	Keep in the Project	Options are: • **Keep in the Project** (include in scenario *all*) • **Delete from Project**
Title			
Background	Enumeration	Gray	Background color of table title.
Foreground	Enumeration	Black	Foreground color of table title.
Default View Options			Default options when showing the tree.
Check Virtual Nodes	Boolean	True	**View** > **Tree Options** > **Check Virtual Nodes**.
Default Project Orientation	Enumeration	horizontal	**View** > **Flow Orientation**.
Display Tool Labels	Boolean	False	**View** > **Table Options** > **Show Tool Labels**.
Show Experiment Numbers	Boolean	True	**View** > **Table Options** > **Show Experiment Numbers**.
Hint Tool Labels	Boolean	False	**View** > **Tree Options** > **Hinting Tool Labels**.

Table 14 Table preferences (Continued)

Preference	Type	Default	Description
Display Additional Node Data	Enumeration	Parameter Values	Options are: • **Parameter Values**: Choose **View** > **Tree Options** > **Parameter Values**. • **Node Numbers**: Choose **View** > **Tree Options** > **Node Numbers**. • **Host**: Choose **View** > **Tree Options** > **Host**. • **Date**: Choose **View** > **Tree Options** > **Date**. • **Execution Time**: Choose **View** > **Tree Options** > **Execution Time**. • **Variables**: Choose **View** > **Tree Options** > **Variables**. • **Job Identifier**: Choose **View** > **Tree Options** > **Job Identifier**.
Show Parameter Tag Names	Boolean	false	**View** > **Table Options** > **Show Parameter Process Names**.
Display Comments	Boolean	false	**View** > **Table Options** > **Show Comment**.
Show Exp Plan	Boolean	false	**View** > **Tree Options** > **Show Experimental Plan**.
Show Info Titles	Boolean	false	**View** > **Table Options** > **Show Information Titles**.
Show Node Numbers	Boolean	false	**View** > **Tree Options** > **Show Node Numbers**.
Show Parameters	Boolean	false	**View** > **Tree Options** > **Show Parameters**.
Show Parameter and Variable Names	Boolean	false	**View** > **Table Options** > **Show Parameter and Variable Names**.
Show Pruned	Boolean	true	**View** > **Tree Options** > **Show Pruned**.
Show Tool Icons	Boolean	true	**View** > **Table Options** > **Show Tool Icons**.
Show Tree	Boolean	true	**View** > **Tree Options** > **Show Tree**.
Show Variables	Boolean	true	**View** > **Tree Options** > **Show Variables**.

Table 14 Table preferences (Continued)

Preference	Type	Default	Description
Show Merged Cells	Boolean	true	**View** > **Tree Options** > **Show Merged Cells**.
Automatically Update Variables	Boolean	false	**Variables** > **Automatically Update Variables**.

Table 15 Scheduler preferences

Preference name	Type	Default	Description
General Settings			General settings of the Sentaurus Workbench scheduler.
CPUs Auto-Detection Upper Limit	Integer	0	Restricts the number of threads detected by the auto-detection algorithm for shared-memory parallelization to the value specified. If 0, the detected number of threads is not limited.
Default Concurrency of Running Experiments	Enumeration	Unlimited	Specifies concurrency running mode for experiments. Options are: • **Unlimited**: Sentaurus Workbench runs all experiments concurrently. • **Limited**: Sentaurus Workbench runs a limited number of experiments concurrently. You can set the limit in the Run Project dialog box.
Nodes Running Order Under Run Limits	Enumeration	Breadth_First	Specifies the order of running nodes when run limits are active: • **Breadth_First**: Sentaurus Workbench tries to complete an entire parametric step of the simulation. • **Depth_First**: Sentaurus Workbench tries to complete an entire experiment before starting another one.
Run Epilogue on Failed and Aborted Nodes	Enumeration	No	Specifies whether to launch an epilogue Tcl script if the simulation job failed or was terminated. Options are **Yes** or **No**. Useful for debugging purposes

Table 15 Scheduler preferences (Continued)

Preference name	Type	Default	Description
Delay After Simulation (msec)	Integer	0	Specifies an optional delay between simulation jobs in milliseconds (**0**: no delay). Useful for slow network file systems.
Protecting of Done Nodes:			
Allow Rerun Done Nodes	Enumeration	Yes	Specifies whether already executed nodes with the *done* status can be rerun.
Show Warning if No Rerun Allowed	Enumeration	No	Specifies whether to show a warning message if users try to run already executed nodes with the *done* status. This option takes effect only if these nodes are not allowed to be rerun.
Update Interval of GUI Scheduler Tab (msec)	Integer	15000	Idle task frequency for updating node statuses in the Sentaurus Workbench user interface.
Local Jobs			Settings applied to locally submitted jobs.
Job Polling interval (msec)	Integer	1000	The interval between sequential checks of the status of the running job (in milliseconds).
Maximum Number of Simultaneous Jobs	Integer	1	Maximum number of jobs that can be launched on a local machine simultaneously.
Default Nice Level	Integer	19	The priority of locally submitted jobs (an argument for the UNIX `nice` command). The range is from 0 to 19.
Check SSH Connectivity for MPI Hosts	Enumeration	Yes	Specifies whether to check given remote hosts for a silent SSH login before launching an MPI job.
LSF Jobs			Setting applied to jobs submitted to LSF.
Job Polling interval (msec)	Integer	1000	The interval between sequential checks of the status of the running job (in milliseconds).

Table 15 Scheduler preferences (Continued)

Preference name	Type	Default	Description
Launcher for MPI Processes	Enumeration	lsf	Sets the launcher that the `mpiexec.hydra` MPI process manager uses to launch MPI worker processes on allocated cluster hosts. Options are: • **lsf**: Launch remote processes using internal LSF mechanism (`blaunch`). • **ssh**: Launch remote processes using SSH. Requires configuring a silent SSH login.
Check SSH Connectivity for MPI Hosts	Enumeration	Yes	Specifies whether to check allocated cluster hosts for a silent SSH login before launching MPI jobs. Takes effect if the launcher for MPI processes is set to **ssh**.
Project Name Command-Line Option	String	-J	Creates the `bsub` command in LSF with the `-P` or `-J` option. The same option also is applied to the `bjobs` command.
Add Tool Name to Project Name	Enumeration	No	Specifies whether Sentaurus Workbench extends the project name with the name of the simulation tool. Takes effect only for the `-J` project name command-line option. If set to **Yes**, then you can see tool names in the output of the LSF `bjobs` command.
SGE Jobs			Setting applied to jobs submitted to Oracle Grid Engine and Univa Grid Engine, as well as other SGE derivatives.
Job Polling interval (msec)	Integer	1000	The interval between sequential checks of the status of the running job (in milliseconds).
Parallel Environment for MPI Jobs	String	dp	Specifies the name of the parallel environment for distributed processing parallelization jobs. Contact your IT for the correct name of this parallel environment. If it is not configured in the SGE grid, then you must configure it (see Appendix G on page 341).

Table 15 Scheduler preferences (Continued)

Preference name	Type	Default	Description
Parallel Environment for SMP Jobs	String	mt	Specifies the name of the parallel environment for shared-memory parallelization (SMP) (multithreaded) jobs. Contact your IT for the correct name of this parallel environment. If it is not configured in the SGE grid, then you must configure it (see Appendix G on page 341).
Launcher for MPI Processes	Enumeration	sge	Sets the launcher that the `mpiexec.hydra` MPI process manager uses to launch MPI worker processes on allocated grid hosts. Options are: • **sge**: Launch remote processes using internal SGE mechanism (`qrsh`). • **ssh**: Launch remote processes using SSH. Requires configuring a silent SSH login.
Check SSH Connectivity for MPI Hosts	Enumeration	Yes	Specifies whether to check the allocated cluster hosts for a silent SSH login before launching MPI jobs. Takes effect if the launcher for MPI processes is set to **ssh**.
TM Jobs			Setting applied to jobs submitted to the TORQUE Resource Manager.
Job Polling interval (msec)	Integer	1000	The interval between sequential checks of the status of the running job (in milliseconds).
RTDA Jobs			Setting applied to jobs submitted to the NetworkComputer.
Job Polling interval (msec)	Integer	1000	The interval between sequential checks of the status of the running job (in milliseconds).

Table 16 Visualization preferences

Preference name	Type	Default	Description
Default S-Visual Interface	Enumeration	Last Used	Sets where to visualize selected files. Options are: • **Last Used**: In the last-used Sentaurus Visual instance. • **Last Created**: In the last-launched Sentaurus Visual instance. • **New Instance**: In a new Sentaurus Visual instance.
File Filters			Predefined file filters available from the **Visualize** > **Sentaurus Visual (Select by Type)** menu.
Boundary	String	`*bnd.tdr`	All Boundary Files.
Mesh	String	`*msh.tdr`	All Mesh Files.
TDR	String	`*.tdr`	All TDR Files.
XY-Plot	String	`*.plt, *.plx`	All XY-Plot Files.
Run Selected Visualizer Nodes Together			Settings applied to the running of selected nodes when choosing **Visualize** > **Run Selected Visualizer Nodes Together**.
Always Preprocess Nodes	Enumeration	No	Specifies whether Sentaurus Workbench preprocesses selected nodes before visualization.

Table 17 Node Explorer preferences

Preference name	Type	Default	Description
Always in Foreground	Enumeration	No	Specifies whether to display the Node Explorer in the foreground, on top of the Sentaurus Workbench window,
Line Numbering	Enumeration	Yes	Switches on or switches off the display of line numbers in the text preview area of the Node Explorer.
Markable Line	Enumeration	Yes	Specifies whether lines can be marked by clicking the line numbers.

Table 18 Utilities preferences

Preference name	Type	Default	Description
PDF Reader	File	evince	Location of the tool for viewing PDF files.
Internet Browser	File	firefox	Location of browser for viewing HTML files,
Diff Tool	File	tkdiff	Location of diff tool executable file.
Editor	File	gedit	Location of text editor executable file. Sentaurus Workbench launches it for both viewing and editing text files. The default text editor is detected as follows: • gedit is used in a GNOME desktop environment. • Kate is used in a KDE desktop environment. • If neither of these text editors is installed, then sedit is used as the default.
Spreadsheet Application	File	oocalc	Location of spreadsheet executable file or tool to use when choosing **Experiments** > **Export** > **Run Spreadsheet Application**.
Image Viewer	File	eog	Location of image viewer. Sentaurus Workbench launches it for viewing `.gif` and `.png` node files generated by Sentaurus Visual.
Layout Viewer	File	none	Location of layout viewer. If **none** is specified, then no layout viewer will be launched.
Terminal	File	xterm	Location of terminal executable file. Sentaurus Workbench launches it when opening a command prompt in a project directory as a separate shell.

Table 19 Behavior on Inactivity

Preference name	Type	Default	Description
Inactivity Time-Out for SWB (hours)	Number	0	Sentaurus Workbench exits or releases license automatically after the specified idle time has elapsed. If 0, no automatic action is applied.
Alert Before Time-Out for SWB (min)	Number	0	Sentaurus Workbench opens a warning dialog box a given number of minutes before it exits automatically. You can cancel the automatic exit by clicking the mouse or pressing a key in the Sentaurus Workbench window. If set to 0, then no warning dialog box opens. This option takes effect only if **Inactivity Time-Out for SWB** is not 0 and **SWB Action After Time-Out** is **Exit**.
SWB Action After Time-Out	String	Exit	Specifies what action Sentaurus Workbench executes after time-out. Options are: • **Exit** • **Release License**
Inactivity Time-Out for Visualization Tools (hours)	Number	0	Sentaurus Visual exits automatically after the specified idle time has elapsed. If 0, no automatic exit is activated.
Alert Before Time-Out for Visualization (min)	Number	0	Sentaurus Visual opens a warning dialog box a given number of minutes before it exits automatically. You can cancel the automatic exit by clicking the mouse or pressing a key in the Sentaurus Visual window. If set to 0, no warning dialog box opens. This option takes effect only if **Inactivity Time-Out for Visualization Tools** is not 0 and **Visualization Tool Action After Time-Out** is **Exit**.
Visualization Tool Action After Time-Out	String	Exit	Specifies what action Sentaurus Visual executes after time-out. Options are: • **Exit** • **Release License**

Table 20 Miscellaneous preferences

Preference name	Type	Default	Description
Attach Applications Library on Start Up	Boolean	true	Attaches Applications Library when Sentaurus Workbench opens.
Attached Roots on Start Up			
Root 1 to Root 5	Directory (subentries)	Path to Applications Library	Attached roots when Sentaurus Workbench opens. Up to five roots are supported.
Number of Bookmarks	Integer	10	Number of last projects visited in Project > Recent Projects. The range is from 0 to 200.
Manuals Front Page	File	Path to manuals	Location of front page for **Help** > **Manuals**.
Idle Task Frequency			Idle task frequency for refreshing (milliseconds).
Automatically Calculate Refresh Rate	Boolean	true	When true, calculates the refresh rate automatically.
Explorer	Nonnegative integer	30000	Projects browser.
Node Refresh Rate	Nonnegative integer	10000	Status of nodes (when running).
Sentaurus Device Material Database	Directory	–	Path to the `MaterialDB` file.
Sentaurus Device Wizard		No	Specifies whether to activate the Sentaurus Device Wizard menu command.
Tailing File Size	Integer	1024	Specifies the maximum size of a node output file, which should be displayed without tailing (in kilobytes). If this size is exceeded, then only the tail of the file is displayed in Sentaurus Workbench.
Tailing Lines Number	Integer	1000	Specifies the number of lines of a node output file to be displayed in the tailing mode.

Table 20 Miscellaneous preferences (Continued)

Preference name	Type	Default	Description
Temp Directory	Directory	`STDB/tmp`	Location of the temporary directory of Sentaurus Workbench.
Training Documentation	File	–	Path to TCAD Sentaurus Tutorial.

Tool Databases

All tools are defined in the tool databases of Sentaurus Workbench. The tool databases (`tooldb`) are global, site, user, and project.

To access a tool database in Sentaurus Workbench, choose the appropriate command:

- **Edit** > **Tool DB** > **Global**
- **Edit** > **Tool DB** > **Site**
- **Edit** > **Tool DB** > **User**
- **Edit** > **Tool DB** > **Project**

Entries in the user tool database complement or override entries in the global and site tool databases, and entries in the project tool database complement or override entries in the other tool databases. The hierarchical order is (in descending order):

1. Project tool database
2. User tool database
3. Site tool database
4. Global tool database

Simulation tools are divided into categories as shown in Figure 31 on page 93. The order of these categories (from top to bottom) reflects the order in which the corresponding simulation phases link to each other (apart from the utilities). For example, it is not typical for a grid generation tool to precede a process simulation tool in the tool flow.

Configuring Tool Databases

The global tool database is usually set up by the systems administrator and is not writable. Initially, the global tool database contains the complete set of TCAD simulation tools and an example of the definition of a *user tool* such as `mytool`.

If your company has multiple groups of users of the TCAD Sentaurus platform, distributed in different sites, it might be useful to customize global database settings for all users at a specific site. To do this, the site tool database must be created. The site tool database is optional.

To create the site tool database:

1. Place your site-specific tool database settings into the `tooldb.tcl` file stored under an arbitrary directory.
2. Set up the following environment variable to refer to this directory:

```
% setenv SWB_SITE_SETTINGS_DIR <path_to_site_directory>
```

Like the global tool database, the site tool database is usually not writable.

You can add your own tools or modify existing ones by changing the user or project tool database.

The global, site, and user tool databases are loaded when starting Sentaurus Workbench or any utilities such as `gsub` and `gjob`. You should restart Sentaurus Workbench to ensure that your changes in these tool databases take effect.

The project tool database is loaded with the project and is applied only to that project. You should reload the project in Sentaurus Workbench to ensure that your changes take effect.

User functions can be defined in the user and project tool databases. These functions can customize standard Sentaurus Workbench preprocessing.

For example, you can extend the standard value extraction algorithm (see Extracted Variables on page 170). First, create your own extract function in the user or project tool database where you traverse the output file `from_file` of the node `nkey` stored in the node directory `nodedir`, and extract those values that are not covered by the standard Sentaurus Workbench value extraction.

The function returns a Tcl list of pairs (name and value). For example, the function can look like:

```
proc ::my_extract_func { nodedir from_file nkey } {
   # extract results from input file
   set extracted_results [list]
   set in_file [file join $nodedir $from_file]
   set in_strm [open $in_file r]
   while { [eof $in_strm] == 0 } {
      set line [gets $in_strm]
      # here you look for names and values to extract
      # ...
      # extracted variable "myVar" has the value stored in the
      # variable "myVal"
      lappend extracted_results [list myVar $myVal]
   }
```

```
   return $extracted_results
}
```

Then, in the same tool database file, redefine the tool epilogue for the required tool. In the case of Sentaurus Topography 3D, it would look like:

```
set WB_tool(sptopo3d,epilogue) { extract_vars "$nodedir" @stdout@
   @node@ ::my_extract_func }
```

The results extracted by these functions are added to the results extracted by the standard Sentaurus Workbench extraction algorithm. However, if you want to overwrite the default extraction, include an additional Boolean flag at the end of the `extract_vars` call in the tool epilogue:

```
set WB_tool(sptopo3d,epilogue) { extract_vars "$nodedir" @stdout@
   @node@ ::my_extract_func true }
```

In this case, only the results that the function extracts appear in Sentaurus Workbench.

Project Tool Database

Sentaurus Workbench handles the project tool database in a different way to other tool databases. Global, site, and user tool databases are sourced in a global Tcl namespace. The project tool database is sourced in a separate Tcl namespace for each loaded project.

Use the following technique when configuring a project tool database:

- Do not globalize Tcl variables (for example, `WB_tool`) with the `global` keyword or the double-colon (`::`) prefix. Otherwise, your project-specific settings might affect other projects.
- Provide the `::` prefix for user-defined functions in a project tool database:

  ```
  proc ::my_extract_funv {nodedir from_file nkey}
  ```

- Use the `globalexists` function to check whether a Tcl variable has been set up on a higher hierarchical level of the tool database (global, site, user):

  ```
  if { [globalexists WB_tool(sdevice,input)] } {
     ...
  }
  ```

- Use the `getglobal` function to access the value that has been set up on a higher hierarchical level of the tool database (global, site, user):

  ```
  set global_tools [getglobal WB_tool(all)]
  ```

- Use the `getactual` function to access the value that has been set up in a project tool database or in a higher hierarchical level of the tool database:

  ```
  set project_tools [getactual WB_tool(all)]
  ```

Note:
It is better to use the `getactual` function rather than the `getglobal` function. The `getactual` function returns the actual value that takes into account all hierarchies of the tool database.

Run Limits Settings

Run limits are switched off by default, which means that no restrictions are applied. To switch on run limits, they must be specified on the global, site, or user levels in XML-compatible files.

The systems administrator typically sets up the file `runlimits.xml` for the global run limits settings, which is not writable.

When companies have multiple groups of users of TCAD Sentaurus, distributed over different sites, it might be useful to customize global run limits settings for all users at a specific site. To do this, the *site run limits file* must be created. This file is optional.

To create the site run limits file:

1. Place your site-specific run limits settings into the `runlimits.xml` file stored under an arbitrary directory.
2. Set up the following environment variable to refer to this directory:

```
% setenv SWB_SITE_SETTINGS_DIR <path_to_site_directory>
```

Like the global run limits settings, the site run limits settings are usually not writable.

The *user run limits settings* complement the ones defined on the global and site levels. The site run limits settings complement the ones defined on the global level. The hierarchical order is (in descending order):

1. User run limits: `$STDB/runlimits_<user>.xml`
2. Site run limits: `$SWB_SITE_SETTINGS_DIR/runlimits.xml`
3. Global run limits: `$STROOT/tcad/$STRELEASE/lib/glib2/runlimits.xml`

When a part of the run limits settings is defined on more than one level, it is overridden in accordance with the hierarchical order.

You can completely or partially prohibit the overriding of run limits settings on the user and site levels by defining a corresponding flag in the global run limits settings.

To access a run limits settings, choose the appropriate command:

- **Edit** > **Run Limits** > **Global**
- **Edit** > **Run Limits** > **Site**
- **Edit** > **Run Limits** > **User**

You can define different types of run limits for each tool:

- Constant run limits, without any specific day or time restriction
- Weekday run limits (Mon–Fri)
- Weekend run limits (Sat, Sun)
- Run limits on specific days of the week (Mon, Tue, Wed, Thu, Fri, Sat, Sun)
- Run limits on special dates (such as public holidays and vacations)

Note:

Timetable run limits settings, where the run limits depend on the day of the week, the time of the day, holidays, and so on, do not support multiple time zones.

Run limits must be defined separately for each tool over which you want to have run limits control. Any combination of different types of run limits can be used. You can bind an arbitrary number of time frames with different run limits for each tool. The only exception is constant run limits: No time frames are allowed for this type of run limits.

Sentaurus Workbench prioritizes the run limits settings in the following way (1=lowest priority and 4=highest priority):

1. Constant run limits
2. Weekday, weekend run limits
3. Run limits on specific days of the week
4. Run limits on special dates

Optional project run limits are defined in Sentaurus Workbench when launching the project (see Defining Run Limits on page 197). Sentaurus Workbench stores the project run limits in a plain text file (`runlimits.txt`) in the project directory.

The file has a very simple structure:

```
sprocess 5

snmesh 5

sdevice 3
```

Only those tools that have nonzero project run limits are stored in the file. You can change the project run limits directly in the `runlimits.txt` file before you launch a project in Sentaurus Workbench or on the command line using the `gsub` command. However, changes to the file will not take effect if the project is already running.

Format of XML-Compatible Run Limits Settings File

Table 21 describes the available tags, their attributes, and the attribute value types for specifying run limits settings.

Note:

The attribute names and values are case sensitive. All values must be enclosed in double quotation marks.

Table 21 Available tags, their attributes, and attribute value type

Tag	Attribute	Attribute value type [Predefined values]	Description
`RunLimitsTable`	`Version`	String `[1.0]`	Attribute used to check which tags or attributes can be read. Ensure you use version `1.0`.
	`Editable`	String `[true, false]`	If `false`, prohibits changing any run limits settings on the next levels (site, user). Default: `true`
	`RestrictionModel`	String `[per_user, none]`	If `none`, switches off a run limits quota, so that no limits are applied. If `per_user`, a run limits quota is applied to all user projects. Default: `per_user`
	`MaxTrials`	Integer	Specifies the maximum number of attempts to acquire a running slot. Default: `17280`
	`PollInterval`	Integer	Defines the time interval, in milliseconds, between two sequential attempts to acquire a running slot. Default: `5000` ms (5 s)

Table 21 Available tags, their attributes, and attribute value type (Continued)

Tag	Attribute	Attribute value type [Predefined values]	Description
	`ProceedAnyway`	String `[true, false]`	Instructs Sentaurus Workbench how to proceed when the maximum waiting time has been reached but no running slot is available. If `false`, the job is considered to have failed. If `true`, the execution of the job proceeds. Default: `false`
	`ApplyToViewers`	String `[true, false]`	Instructs Sentaurus Workbench to apply run limits to both batch and interactive sessions of the tool launched from within Sentaurus Workbench. Default: `false`
`Notification`	`MaxRunTime`	Integer	Defines the runtime limits in hours for all tools. A job that runs longer is considered a long-running job. If zero, runtime limits are switched off. Default: `0`
	`Interval`	Integer	Defines the time interval in minutes between two sequential notifications for a long-running job. Default: `0`
	`MaxTrials`	Integer	Defines the maximum number of notifications to be sent before the final action. Default: `0`
	`Type`	String `[Email, EmailAndLog, Log]`	Defines the type of notification. Default: `Log`
	`Recipient`	String	Defines the user name for email notification. If empty, sends an email to the current user. Default: empty
	`Action`	String `[Abort, Nothing]`	Defines the action to perform after the final notification. Default: `Nothing`
`Tool`	`Name`	String	Name of tool.

Table 21 Available tags, their attributes, and attribute value type (Continued)

Tag	Attribute	Attribute value type [Predefined values]	Description
`RunLimit`	`Value`	Integer	Defines the maximum number of simultaneously running instances of a tool.
`SubmitDelay`	`Value`	Integer	Defines the interval in seconds for sequential submission of two instances of a tool.
`TimeSegment`	`Begin`	Time: HH:MM	Time in 00:00–24:00[1] format.
	`End`	Time: HH:MM	Time in 00:00–24:00[1] format. Must be later than `Begin`. Otherwise, values will be swapped.
`Holiday`	`Begin`	Date: DD/MM/[YYYY]	Date in 10/01/2022 format. Year is optional; if not defined, it means *every year*.
	`End`	Date: DD/MM/[YYYY]	Date in 28/01/2022 format. Year is optional; if not defined, it means *every year*.
	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Weekdays`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Weekends`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Mon`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Tue`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Wed`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Thu`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)

Table 21 Available tags, their attributes, and attribute value type (Continued)

Tag	Attribute	Attribute value type [Predefined values]	Description
`Fri`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Sat`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)
`Sun`	`Editable`	String `[true, false]`	Prohibits (`false`) or allows (`true`) tag editing. Default: `true`)

1. For `TimeSegment` tag, only one attribute is mandatory, either `Begin` or `End`. If the second attribute is omitted, the missing value is generated as follows: 00:00 for `Begin` attribute and 24:00 for `End` attribute.

You must follow these rules in run limits XML settings:

- No empty lines in the file.
- For timetable settings, time segments must not intersect the end of the day. Use 00:00-08:00, 08:00-17:00, or 17:00-24:00 rather than 08:00-17:00 or 17:00-08:00.
- The `Weekends` or `Weekdays` tag is mandatory for timetable settings.

Example 1

Assume you want to define the following run limits for Sentaurus Process (`sprocess`), which can be redefined on a user level:

Weekdays:
08:00-17:00: run limit = 10
17:00-22:00: run limit = 30

Nights, weekends, the rest of time: run limit = 80

The following timetable is applicable:
General run limits for tool "sprocess": 80
Weekdays (Mon-Fri):
08:00-17:00: run limit = 10
17:00-22:00: run limit = 30

This is the run limits settings file that would implement these specifications:

```
<RunLimitsTable Version="1.0" RestrictionModel="per_user">
   <Tool Name="sprocess">
      <RunLimit Value="80" />
      <Weekdays>
```

```
          <TimeSegment Begin="08:00" End="17:00">
             <RunLimit Value="10" />
          </TimeSegment>
          <TimeSegment Begin="17:00" End="22:00">
             <RunLimit Value="30" />
          </TimeSegment>
       </Weekdays>
    </Tool>
</RunLimitsTable>
```

Example 2

Assume you want to implement a more complex run limits timetable for Sentaurus Process (`sprocess`), which cannot be redefined on a user level:

General: run limit = 20

Weekdays: run limit = 20
08:00-17:00: run limit = 5
18:00-24:00: run limit = 30

Weekends: run limit = 40

Monday: 13:00-14:00: run limit = 10
Tuesday: 15:00-17:00: run limit = 15
Wednesday: 17:00-24:00: run limit = 15
Thursday: 15:00-17:00: run limit = 15
Friday: 15:00-24:00: run limit = 35
Saturday: 00:00-07:00: run limit = 50
Sunday: 20:00-24:00: run limit = 5

Exceptions (for example, vacation):
10 January 2022 – 28 January 2022
General: run limit = 15
17:00-24:00: run limit = 5

1 August every year
General: run limit = 60

This is the run limits settings file that would implement these specifications:

```
<RunLimitsTable Version="1.0" Editable="false"
RestrictionModel="per_user">
   <Tool Name="sprocess">
      <RunLimit Value="20" />
      <Weekdays>
         <RunLimit Value="20" />
         <TimeSegment Begin="08:00" End="17:00">
            <RunLimit Value="5" />
         </TimeSegment>
         <TimeSegment Begin="18:00" End="24:00">
```

```
          <RunLimit Value="30" />
       </TimeSegment>
    </Weekdays>
    <Weekends>
       <RunLimit Value="40" />
    </Weekends>
    <Mon>
       <TimeSegment Begin="13:00" End="14:00">
          <RunLimit Value="10" />
       </TimeSegment>
    </Mon>
    <Tue>
       <TimeSegment Begin="15:00" End="17:00">
          <RunLimit Value="15" />
       </TimeSegment>
    </Tue>
    <Wed>
       <TimeSegment Begin="17:00" End="24:00">
          <RunLimit Value="15" />
       </TimeSegment>
    </Wed>
    <Thu>
       <TimeSegment Begin="15:00" End="17:00">
          <RunLimit Value="15" />
       </TimeSegment>
    </Thu>
    <Fri>
       <TimeSegment Begin="15:00" End="24:00">
          <RunLimit Value="35" />
       </TimeSegment>
    </Fri>
    <Sat>
       <TimeSegment Begin="00:00" End="07:00">
          <RunLimit Value="50" />
       </TimeSegment>
    </Sat>
    <Sun>
       <TimeSegment Begin="20:00" End="24:00">
          <RunLimit Value="5" />
       </TimeSegment>
    </Sun>
    <Holiday Begin="10/01/2022" End="28/01/2022">
       <RunLimit Value="15" />
       <TimeSegment Begin="17:00" End="24:00">
          <RunLimit Value="5" />
       </TimeSegment>
    </Holiday>
    <Holiday Begin="01/08">
       <RunLimit Value="60" />
    </Holiday>
 </Tool>
</RunLimitsTable>
```

Example 3

Assume you want to be notified if any simulation job runs longer than 1 day. However, for Sentaurus Device jobs you want to be notified if they run more than 2 days. For each long-running job, you want to receive up to five notifications at 30 minute intervals. After the final notification, you want Sentaurus Workbench to terminate the job.

This is the run limits settings file that would implement these specifications:

```
<RunLimitsTable Version="1.0" RestrictionModel="per_user">
   <Notification>
      <MaxRunTime Value="24" />
      <Interval Value="30" />
      <MaxTrials Value="5" />
      <Type Value="Email" />
      <Recipient Value="jsmith" />
      <Action Value="Abort" />
   </Notification>
   <Tool Name="sdevice">
      <Notification>
         <MaxRunTime Value="48" />
         <Interval Value="30" />
         <MaxTrials Value="5" />
         <Type Value="Email" />
         <Recipient Value="jsmith" />
         <Action Value="Abort" />
      </Notification>
   </Tool>
</RunLimitsTable>
```

Changing the Run Limits Settings

Sentaurus Workbench automatically recognizes any changes you apply to the global, site, or user run limits settings. Changes in the corresponding files apply to both already running projects and projects you will launch after the changes.

You can use this capability when organizing a fair distribution of licenses among your work group. For example, you can implement a script that monitors the availability of licenses on the license server (for example, using the `lmstat` command) and updates the global run limits file accordingly.

Bypassing Unwanted License Checks

Sentaurus Workbench checks the availability of Sentaurus PCM Studio licenses on startup. If licenses are available, Sentaurus Workbench extends the menu bar with additional items (see Table 37 on page 324).

Depending on the setup of the Synopsys Common Licensing (SCL) server, checking the availability of nonexistent licenses might take a considerable amount of time, and you might think that Sentaurus Workbench is not working.

If you do not have Sentaurus PCM Studio licenses and experience a long loading time of Sentaurus Workbench, set the following environment variable to bypass unwanted license checks in Sentaurus Workbench:

```
setenv SWB_BYPASS_AUTH_spcmstd 1
```

If you use Bash instead of C shell, specify:

```
export SWB_BYPASS_AUTH_spcmstd=1
```

10

Integrating Sentaurus Workbench With Other Tools

This chapter discusses how to use other tools and features from Sentaurus Workbench.

Creating Symbolic Links to Node Output Files

In some process simulation projects, it can be useful to look for simulation results not by node number, but by a combination of experiment number, parameter name, or parameter process name. Sentaurus Workbench allows you to create these symbolic links automatically.

Note:

This feature applies only to process tool steps.

To create symbolic links to node output files:

1. Select a process tool step.
2. Choose **Tool** > **Create Process Output Links**.

Note:

The symbolic link is not created when there is a file with the same name.

Process simulators store simulation results in TDR files with the following naming convention:

```
n<node>_<tool_acronym>.tdr
```

As the result of automatic linking, all available node output TDR files are linked using the following symbolic links:

```
<experiment_number>_<parameter>_<process_name>.tdr
```

Note:

Sentaurus Workbench allows a node to belong to multiple experiments. In this case, the first matching experiment number is used to create the symbolic links.

Visualizing Response Surface Models

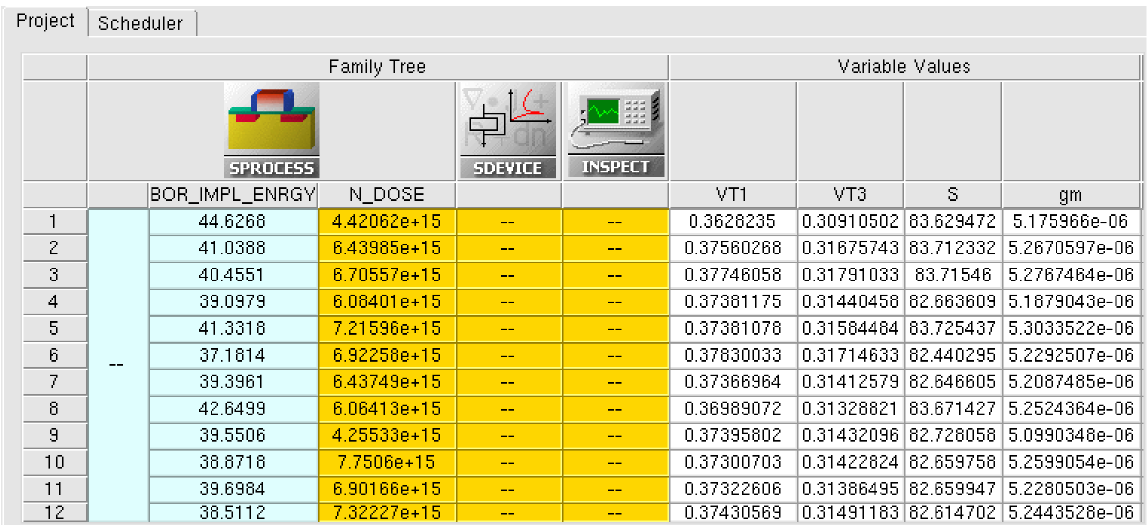
You can visualize a surface corresponding to a response surface model (RSM).

To visualize a response surface model:

1. Run Sentaurus Workbench and open a project containing response values (see Figure 73).
2. Choose **Optimization** > **RSM Visualization**.

   The RSM Visualization window opens (see Figure 74 on page 257).
3. See Step 1 of RSM Visualization.

Figure 73 Example of project with parameterization and responses



	Family Tree					Variable Values			
		BOR_IMPL_ENRGY	N_DOSE			VT1	VT3	S	gm
1	--	44.6268	4.42062e+15	--	--	0.3628235	0.30910502	83.629472	5.175966e-06
2		41.0388	6.43985e+15	--	--	0.37560268	0.31675743	83.712332	5.2670597e-06
3		40.4551	6.70557e+15	--	--	0.37746058	0.31791033	83.71546	5.2767464e-06
4		39.0979	6.08401e+15	--	--	0.37381175	0.31440458	82.663609	5.1679043e-06
5		41.3318	7.21596e+15	--	--	0.37381078	0.31584484	83.725437	5.3033522e-06
6		37.1814	6.92258e+15	--	--	0.37830033	0.31714633	82.440295	5.2292507e-06
7		39.3961	6.43749e+15	--	--	0.37366964	0.31412579	82.646605	5.2087485e-06
8		42.6499	6.06413e+15	--	--	0.36989072	0.31328821	83.671427	5.2524364e-06
9		39.5506	4.25533e+15	--	--	0.37395802	0.31432096	82.728058	5.0990348e-06
10		38.8718	7.7506e+15	--	--	0.37300703	0.31422824	82.659758	5.2599054e-06
11		39.6984	6.90166e+15	--	--	0.37322606	0.31386495	82.659947	5.2280503e-06
12		38.5112	7.32227e+15	--	--	0.37430569	0.31491183	82.614702	5.2443528e-06

Step 1 of RSM Visualization

In Step 1 of RSM visualization (see Figure 74), specify the following fields to define the RSM to be visualized:

Scenario The scenario in the loaded project that will be used to create the RSM.

X-axis The parameter to be used as the x-axis of the plot.

Y-axis The parameter to be used as the y-axis of the plot.

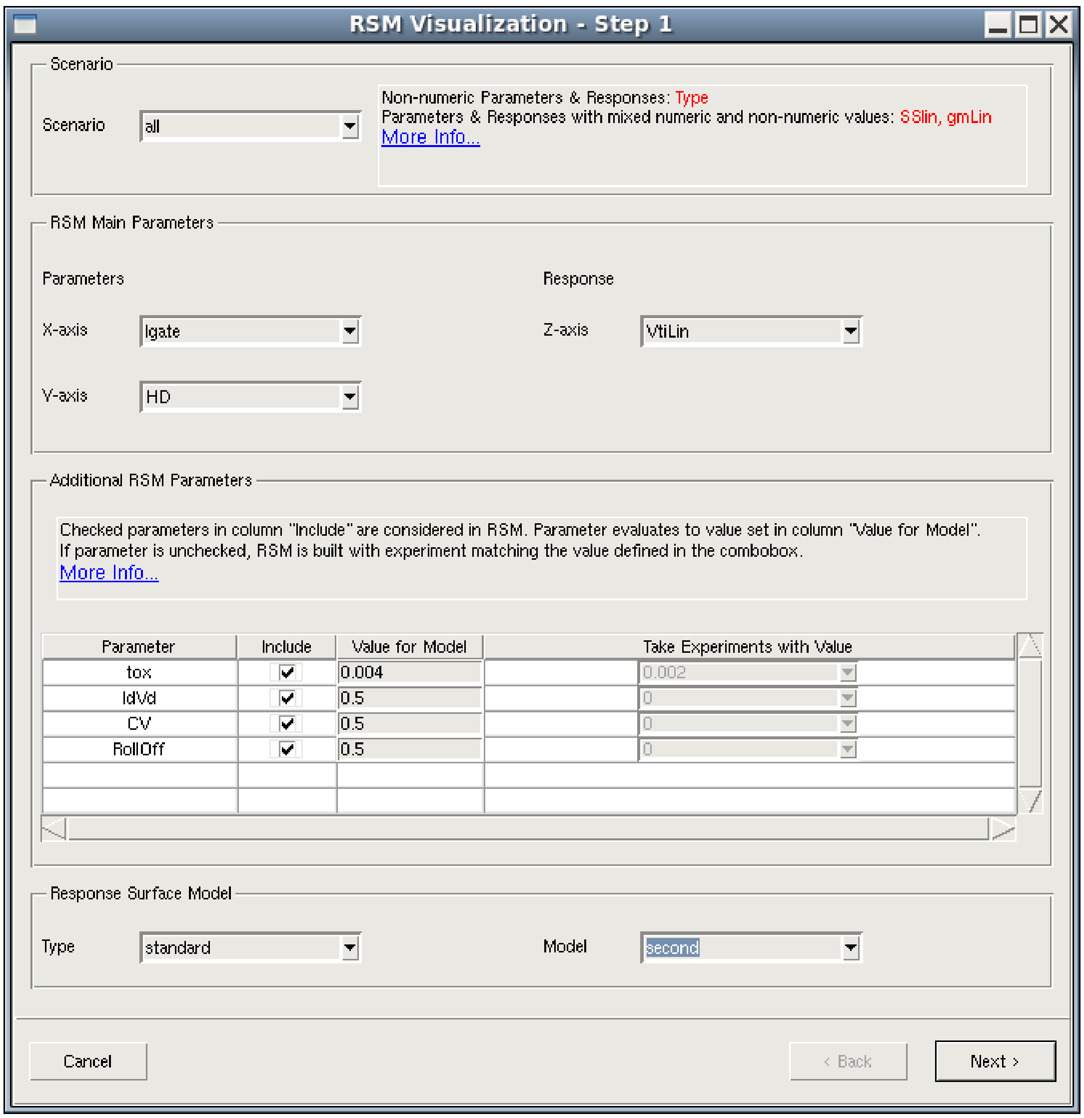
Z-axis The response to be used as the z-axis of the plot.

Type The type of RSM model (standard or Kriging).

Model The RSM model degree (first order or second order).

All project parameters that are not assigned to the x-axis or y-axis can be included in the modeling.

Figure 74 RSM Visualization - Step 1



If any project parameters or responses have any nonnumeric values, Sentaurus Workbench specifies their names in the Scenario group box. You cannot include these parameters and responses in the modeling.

Including Parameters in the Modeling

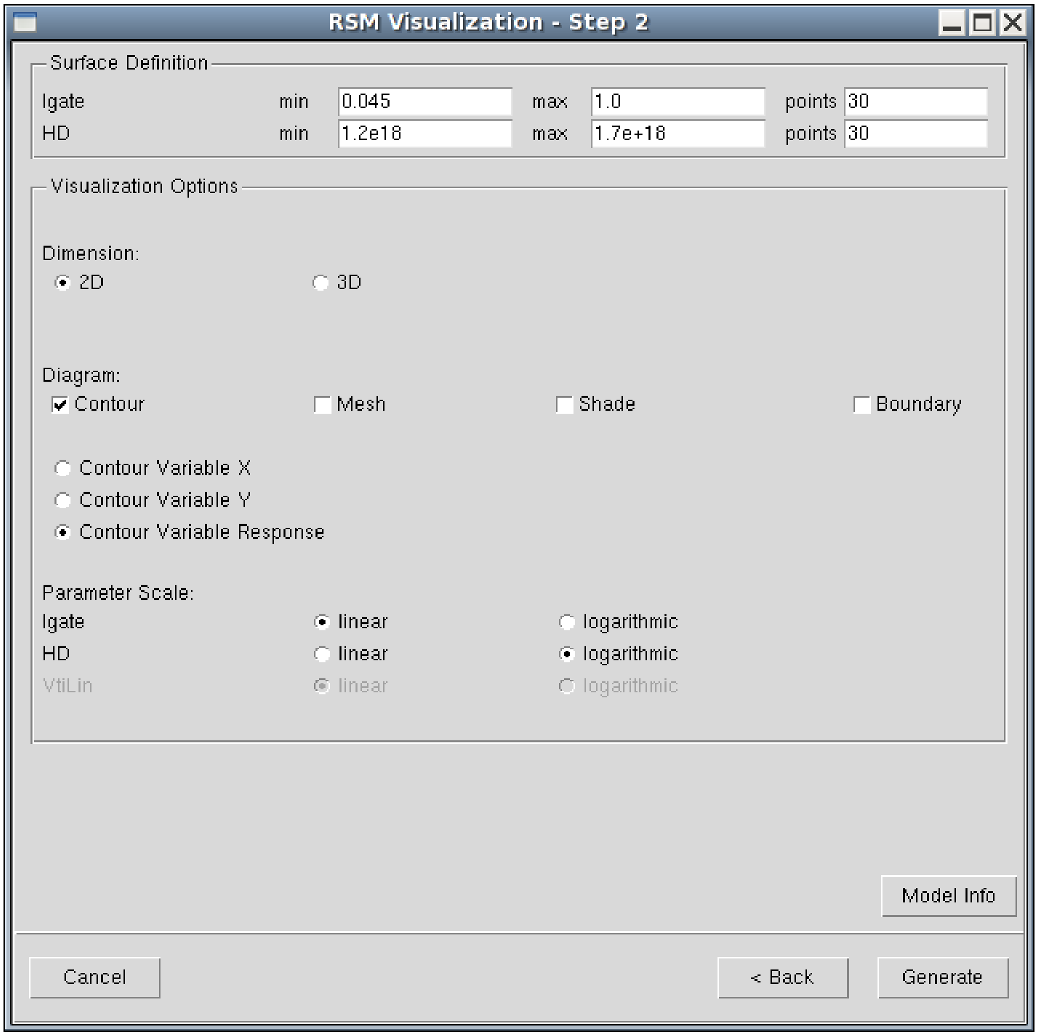
To include parameters in the modeling:

1. Select the corresponding check box in the Include column.
2. Specify the fixed value in the Value for Model column.
3. Click **Next** to continue.
4. See Step 2 of RSM Visualization.

Step 2 of RSM Visualization

In Step 2 of RSM visualization (see Figure 75), select the values that determine the plot ranges and the number of points for the parameters attached to the x-axis and y-axis of the resulting plot. Sentaurus Workbench automatically calculates the minimum and maximum parameter values from the project data. The default value for the number of points is 30.

Figure 75 RSM Visualization - Step 2



Visualization Options

Sentaurus Workbench offers different visualization options to display the RSM.

Table 22 Visualization options

Option	Description
2D	Displays an xy plot when one parameter and a response are selected. When you select two parameters and a response, the response is displayed as a contour plot.

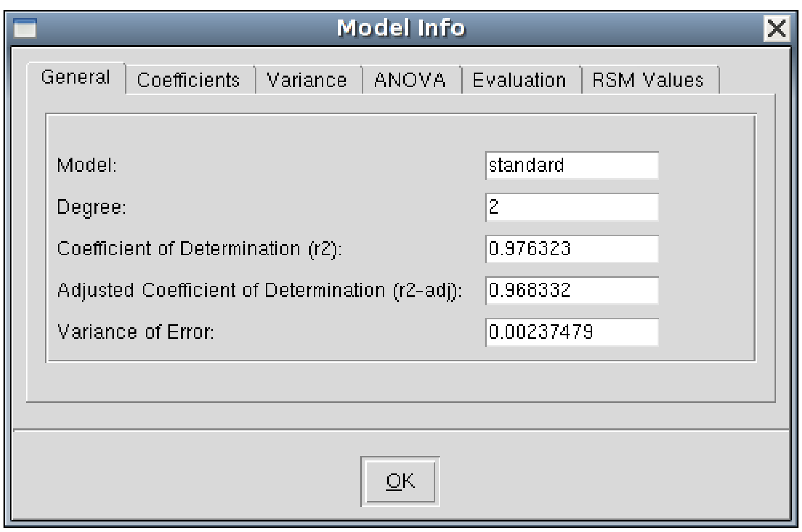
Table 22 Visualization options (Continued)

Option	Description
3D	Displays a 3D surface. This is the default setting when you select two parameters and a response.
Boundary	Sets the plotting style. Displays the boundaries of the surface.
Contour	Sets the plotting style. Displays the surface as a contour plot. When you select this option, you can specify whether x, y, or the response should be used as the contour variable. By default, the response is selected as the contour variable. The parameters and the response can be displayed in a logarithmic or linear scale.
Mesh	Sets the plotting style. Displays the surface as a mesh plot.
Shade	Sets the plotting style. Displays the surface as a shade plot.

Model Information

Click the **Model Info** button to see information about the model (see Figure 76).

Figure 76 Model Info dialog box for an RSM

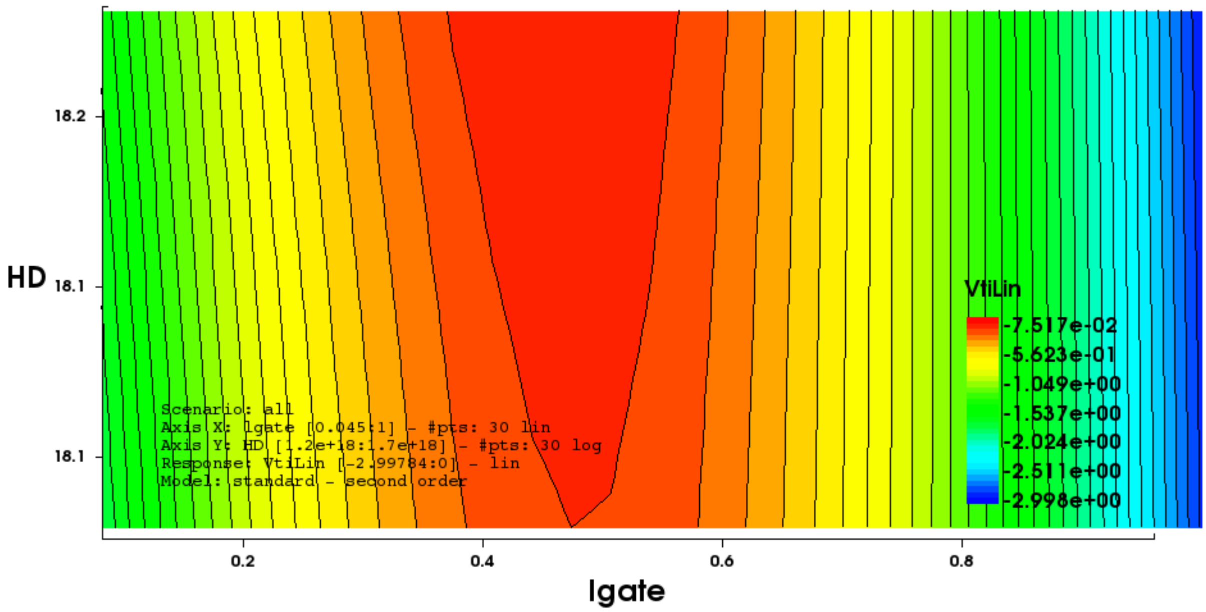


The following information about a model is available: statistics, coefficients, variance, ANOVA table, and evaluation of the experiments that were used to create the RSM. In addition, you can use the **Evaluation** tab to evaluate the model for any combination of parameter values.

Visualizing the Model

To visualize the model, click the **Generate** button (see Figure 75 on page 259).

Figure 77   Example of 2D RSM contour plot visualized in Sentaurus Visual



11

Schedulers

This chapter describes the schedulers used in Sentaurus Workbench.

Scheduling Systems

Parameter studies involving several parameters usually require a large number of simulation runs. The limited interdependency of the individual simulation steps makes this type of application an excellent candidate for exploiting coarse-grain parallelism on a network of workstations.

Sentaurus Workbench offers access to different, more or less sophisticated, backend job scheduling systems. Access to these systems is transparent. However, the installation and initialization of a distributed scheduling system is usually a complicated task, as it requires many conditions to be fulfilled in the network environment. A distributed system is also much more difficult to troubleshoot than a conventional application.

Sentaurus Workbench provides a unified scheduling approach, by which you can submit jobs to any scheduler or queue. Each node can be assigned to any queue defined by the systems administrator in the global queue file or site queue file.

This section provides guidelines to set up, test, and successfully operate the scheduling facilities of Sentaurus Workbench. Details about installing and configuring the IBM Platform LSF system, the Oracle Grid Engine system, the Univa Grid Engine system, the TORQUE Resource Manager system, and the Runtime Design Automation NetworkComputer can be found in the appropriate vendor manuals.

Supported Schedulers

Sentaurus Workbench supports the following backend execution engines:

IBM® Platform™ LSF (Load Sharing Facility)

IBM Platform LSF offers a wide range of scheduling options and a complete set of administration tools. LSF is a corporate-size solution to consolidate computational resources into a cluster. See LSF Scheduler on page 264.

Oracle Grid Engine (formerly, Sun Grid Engine)

The Sun Grid Engine (SGE) was originally developed by Sun Microsystems as an open-source batch-queuing system. Now, Univa supports it as the Oracle Grid Engine. There are several open-source and proprietary derivatives of the Sun Grid Engine, for example, the Univa® Grid Engine™. Such SGE-based solutions are typically used on a computer farm or computer cluster, and are responsible for accepting, scheduling, dispatching, and managing the remote execution of large numbers of standalone, parallel, or interactive user jobs. In addition, it manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses. SGE is better suited to large corporate networks. See SGE Scheduler on page 267.

TORQUE Resource Manager

TORQUE Resource Manager (TM), supported by Adaptive Computing, Inc., provides control over batch jobs and distributed computing resources. It is an advanced open-source product based on the original PBS project and incorporates the best of both community and professional development. See TM Scheduler on page 270.

Runtime Design Automation NetworkComputer™

NetworkComputer (RTDA) is an enterprise-grade high-performance job scheduler, otherwise known as a batch processing system or a distributed resource management system (DRMS). It provides a cost-effective solution for distributing an IT workload over any compute farm (cluster) topology. NetworkComputer can manage farms with many thousands of processors, with instantaneous workloads in excess of 2 million jobs, and throughput of millions of jobs per day. RTDA Scheduler on page 272.

Local

The local scheduler executes jobs sequentially on the local machine. It does not require any special setup and does not produce any scheduling overhead. It is useful for running short simulations, or setting up and debugging more complex projects.

The LSF, SGE, TM, and RTDA systems work on a cluster of heterogeneous workstations sharing a common file namespace. Therefore, the following conditions are mandatory for their proper use:

- Synopsys TCAD simulation software is installed on all of the workstations that will be used. `STROOT` is identical (same absolute path) on all workstations. The easiest way to obtain this is to share (NFS mount) the file system where the Synopsys software is installed.
- The user project space is shared by all participating workstations and accessible through the same absolute path. Again, the easiest way to obtain this is to NFS mount the file system where the `STDB` of the user resides.
- A global queue definition file and, when necessary, a site queue definition file must be set up by the systems administrator, who provides the list containing queue names.

LSF Scheduler

Sentaurus Workbench provides an interface to the Load Sharing Facility (LSF) system, which integrates a cluster of heterogeneous workstations into a single system environment and provides sophisticated job scheduling policies. It can be used to submit and distribute project simulation jobs over the local network.

To use LSF, the following conditions must be fulfilled:

1. The local host (the one running Sentaurus Workbench and `gsub`) is a LSF client or server.
2. On the local host, the LSF binaries are accessible through the environment variable `PATH` (add `$LSF/bin` to your `PATH` variable before starting Sentaurus Workbench).
3. Synopsys TCAD simulation software is accessible to all LSF server workstations.
4. The project space is accessible to all LSF servers with the same absolute path.

To confirm points 1 and 2, the following two commands should succeed on the local host:

```
% lsid
IBM Spectrum LSF Standard 10.1.0.0, Jul 08 2016
Copyright International Business Machines Corp, 1992-2016.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.

My cluster name is snps
My master name is snpsemt317

% bqueues
QUEUE_NAME    PRIO STATUS      MAX JL/U JL/P JL/H NJOBS PEND RUN SUSP
bhigh         100  Open:Active  -    -    -    -   0     0    0   0
vg            100  Open:Active  -    -    -    -   1     1    0   0
normal        100  Open:Active  -    -    -    -   0     0    0   0
nightly_build 100  Open:Active  -    -    -    -   1     1    0   0
ams_cae       100  Open:Active  -    -    -    -   0     0    0   0
bnormal        40  Open:Active  -    -    -    -   28    18   10  0
lost_and_found  1  Closed:Inact -    0    0    -   22    22   0   0
...
```

The `gsub` utility uses the `bsub` command to submit jobs to LSF, the `bjobs` command to track the job status, and the `bkill` command to terminate jobs. You can specify additional `bsub` command-line options using queue options (see Configuring Scheduling Systems on page 273).

To execute each node, `gsub` does not directly submit the corresponding tool command line, but launches a job wrapper, the `gjob` utility. The latter updates the job status and locally executes the prologue tool and epilogue of the corresponding tool as defined in the tool database.

Note:

Tool database settings control which `bsub`, `bjobs`, and `lsid` commands will be executed. The corresponding settings are:

```
WB_binaries(tool,bsub)
WB_binaries(tool,bjobs)
WB_binaries(tool,lsid)
```

You can control which command-line option the LSF command `bsub` uses to link a submitted job to a specific LSF project name or to LSF job name attributes. By default, LSF uses the `-P` option and specifies the LSF project name. Alternatively, you can use the `-J` option, which specifies the LSF job name.

To switch to the `-J` option:

1. Choose **Edit** > **Preferences** or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Scheduler** > **LSF Jobs**.
3. Set **Project Name Command-Line Option** to **-J**.
4. Click **Apply**.

You can specify LSF resource requirements on a per-tool basis. Different resource requirements can be set up for specific tools in a Sentaurus Workbench project. For the syntax of the LSF resource requirement string, refer to the definition of the `-R` option of the `bsub` command in the LSF documentation.

The per-tool LSF resource requirements can be set up in the Sentaurus Workbench tool database (on the global, user, or project level) using the syntax:

```
set WB_tool(<tool_name>,LSF,resource) {<resource_string>}
```

where `<tool_name>` is the database name of the tool, and `<resource_string>` defines the resource requirements. For example:

```
set WB_tool(sprocess,LSF,resource) {rusage[sprocess_all=1,
    sprocess3d_all=1]}
set WB_tool(sprocess,LSF,resource) {rusage[sprocess_all=1|
    |SK_sprocess_all=1]}
set WB_tool(sdevice,LSF,resource) {mem>5000 rusage[sdevice_all=1]}
```

The resource requirement string, specified in the tool database, overwrites the one specified in the LSF queue in the Sentaurus Workbench global or site queue configuration files (`gqueues.dat`).

For example, when the following LSF queue name is defined in the queue configuration file:

```
queue lsf:mylsfqueue "bsub -R \"swp > 5 && mem > 10\" -q normal"
```

and you run a Sentaurus Process simulation over this LSF queue, the tool-related resource setting `rusage[sprocess_all=1,sprocess3d_all=1]` replaces the global setting `swp > 5 && mem > 10`.

Note:

Do not specify the `-R` option for the resource requirement string. Sentaurus Workbench adds it automatically.

Troubleshooting LSF

This section describes how to solve two typical issues you might experience with the LSF cluster.

Nodes Submitted But Not Executed

This issue can appear on overworked or slow LSF clusters. After Sentaurus Workbench submits node jobs in the LSF environment with the `bsub` command, it starts periodical checking of job polling by calling the `bjobs` command to obtain the status of the submitted jobs. In some cases, the first call of `bjobs` occurs before the LSF scheduler completes the actual job submission. This results in an empty output of the `bjobs` command, which Sentaurus Workbench considers to be the end of the LSF execution session.

Workaround

Sentaurus Workbench provides options to control the job-polling algorithm, which you can fine-tune to allow the LSF cluster to work more effectively with Sentaurus Workbench:

- The tool database setting:

  ```
  WB_tool(bjobs,delay)
  ```

  specifies the time interval between the submission of the first node and the first call of the `bjobs` command to check the status of the jobs. Set the value in seconds. By default, Sentaurus Workbench defines it as 15 s.

- The preferences setting **Scheduler** > **LSF Jobs** > **Job Polling Interval (sec)** specifies the time interval between two sequential calls of `bjobs`. This value is set in seconds (default is 1 s). Increasing this time interval might be helpful for LSF clusters that are overworked with many requests.

Nodes Not Executed and Log File Contains Complaints About bjobs Output

This issue can appear when the LSF `bjobs` command provides output in a format that is not expected by Sentaurus Workbench, for example, when a user-specific script or application is called instead of the standard LSF `bjobs` command.

Workaround

You can adapt nonstandard `bjobs` output by defining your own parsing algorithm. Implement the following Tcl function and put it into the tool database:

```
proc ::glsf::ParseBjobsOutput { text } {
   ...
}
```

This Tcl function overwrites the standard one that Sentaurus Workbench uses for parsing the `bjobs` output. The function returns the output of the `bjobs` call as the input (text) and parses it. Sentaurus Workbench expects this function to return the Tcl list of the following data triplets for each LSF job:

- `jobID`
- `status`
- `execution_host`

For example, the returning Tcl list can look like:

```
{ {1234 DONE tcadxps1} {3343 EXIT tcadcat3} {1123 PEND myhost} }
```

Supported status values are:

- `DONE`
- `EXIT`
- `PEND`
- `RUN`
- `PSUSP`
- `USUSP`
- `SSUSP`

SGE Scheduler

As an alternative to the LSF scheduler, Sentaurus Workbench provides an interface to the SGE scheduler. SGE is a queuing facility that manages various system resources, running different operating systems, and helps you to submit and manage your jobs.

To use SGE, the following conditions must be fulfilled:

1. Correct SGE environment settings are specified. Check with your systems administrator on which SGE configuration script to source or how to modify your environment settings

manually. In particular, the SGE binaries are accessible through the environment variable `PATH`.

2. The local host (the one running Sentaurus Workbench and `gsub`) is an SGE submit or admit host.
3. Synopsys TCAD simulation software is accessible to all SGE hosts.
4. The project space is accessible to all SGE hosts with the same absolute path.

To confirm points 1 and 2, the following command should succeed on the local host:

```
% qconf -sconf global
```

This command displays the SGE scheduler configuration. If point 2 is not fulfilled, you will see the following output:

```
% qconf -sconf global
denied: host "myhostname" is neither submit nor admin host
```

In this case, check with your systems administrator.

The `gsub` utility uses the `qsub` command to submit jobs to SGE, the `qstat` command to track the job status, and the `gdel` command to terminate jobs. You can specify additional `qsub` command-line options using queue options (see Configuring Scheduling Systems on page 273). To execute each node, `gsub` does not directly submit the corresponding tool command line, but launches a job wrapper, the `gjob` utility. The latter updates the job status and locally executes the prologue tool and epilogue of the corresponding tool as defined in the tool database.

You can specify SGE resource descriptions on a per-tool basis. Different resource descriptions can be set up for specific tools in a Sentaurus Workbench project. For descriptions of the SGE resource flags, refer to the definition of the `-l` option of the `qsub` command in the SGE documentation.

The per-tool resource descriptions can be set up in the Sentaurus Workbench tool database (on the global, user, or project level) using the syntax:

```
set WB_tool(<tool_name>,SGE,resource) {<resource_description>}
```

where <tool_name> is the database name of the tool, and <resource_description> defines the comma list of resource flags. For example:

```
set WB_tool(sprocess,SGE,resource)
   {arch=glinux,kernel_version=2.6.9,num_proc=4,mem_inst=2G}
set WB_tool(tsuprem4,SGE,resource) {arch=glinux,kernel_version=2.4.21}
```

The resource description, specified in the tool database, overwrites the one specified in the SGE queue in the Sentaurus Workbench global or site queue configuration files (`gqueues.dat`). For example, when the following SGE queue name is defined in the queue configuration file:

```
queue sge:normal64 "qsub -V -cwd -now n -P bnormal -l
   arch=glinux,kernel_version=2.6.9"
```

and you run a Sentaurus Process simulation over this SGE queue, the tool-related resource description `arch=glinux,kernel_version=2.6.9,num_proc=4,mem_inst=2G` replaces the global setting `arch=glinux,kernel_version=2.6.9`.

Note:

Do not specify the `-l` option for the resource requirement string. Sentaurus Workbench adds it automatically.

Before setting up resource descriptions in the tool database and the global or site queue file, it is recommended to check the validity of the descriptor and the availability of the requested resources. You can do this by using the following command:

```
% qhost -l <resource_descriptor>
```

Other useful SGE commands are:

```
% qhost
```

Returns the list of SGE hosts that fulfill your resource descriptor.

```
% qaccess
```

Allows you to check which SGE hosts you have access to.

```
% qconf -sc
```

Provides you with detailed information on the resource flags that you can use in your resource descriptors.

```
% qconf -se <SGE_hostname>
```

Informs you on the resource flags that are supported by the given SGE host.

Troubleshooting SGE

Systems administrators of overworked SGE clusters might receive complaints from Sentaurus Workbench users about job polling occurring too frequently. This section describes how to fine-tune Sentaurus Workbench schedule settings in this case.

Job Polling Occurs Too Frequently

This issue can occur on overworked or slow SGE clusters. After Sentaurus Workbench submits node jobs in the SGE environment with the `qsub` command, it starts a periodical check of job polling by calling the `qstat` command to obtain the status of the submitted jobs.

Workaround

You can reduce this frequency in the Sentaurus Workbench preferences and the tool database.

The preferences setting **Scheduler** > **SGE Jobs** > **Job Polling Interval (sec)** specifies the time interval (in seconds) between two sequential calls of `qstat` (default: 1 s). Increasing this time interval might be helpful for SGE clusters that are overworked with many requests.

TM Scheduler

As an alternative to LSF and SGE, Sentaurus Workbench provides an interface to the TORQUE Resource Manager (TM) scheduler. This scheduler is an advanced open-source product and incorporates the best of both community and professional development. It incorporates significant advances in the areas of scalability, reliability, and functionality.

TM can be freely used, modified, and distributed, which is an important advantage over LSF and proprietary derivatives of SGE.

To use TM, the following conditions must be fulfilled:

1. Correct TM environment settings are set up. Check with your systems administrator about which TM configuration script to source or how to modify your environment settings manually. In particular, the TM binaries are accessible through the environment variable `PATH`.
2. The local host (the one running Sentaurus Workbench and `gsub`) is a TM submit or admit host.
3. Synopsys TCAD simulation software is accessible to all TM hosts.
4. The project space is accessible to all TM hosts with the same absolute path.

To confirm points 1 and 2, the following command should succeed on the local host:

```
% qstat -u non_existing_user
```

If this command does not work properly, check with your systems administrator.

The `gsub` utility uses the `qsub` command to submit jobs to TM, the `qstat` command to track the job status, and the `qdel` command to terminate jobs. You can specify additional `qsub` command-line options using the queue options (see TM Queues on page 277). To execute each node, `gsub` does not directly submit the corresponding tool command, but launches a job wrapper, the `gjob` utility. The latter updates the job status and locally executes the prologue tool and the epilogue of the corresponding tool as defined in the tool database.

You can specify TM resource descriptions on a per-tool basis. Different resource descriptions can be set up for specific tools in a Sentaurus Workbench project. For descriptions of the TM resource flags, refer to the definition of the `-l` option of the `qsub` command in the TORQUE Resource Manager documentation.

The per-tool resource descriptions can be set up in the Sentaurus Workbench tool database (on the global, user, or project level) using the syntax:

```
set WB_tool(<tool_name>,TM,resource) {<resource_description>}
```

where `<tool_name>` is the database name of the tool, and `<resource_description>` defines the comma list of resource flags.

Note:

Do not specify the `-l` option for the resource requirement string. Sentaurus Workbench adds it automatically.

The resource description, specified in the tool database, overwrites the one specified in the TM queue in the Sentaurus Workbench global or site queue configuration files (`gqueues.dat`).

Troubleshooting TM

Systems administrators of overworked TM clusters might receive complaints from Sentaurus Workbench users about job polling occurring too frequently. This section describes how to fine-tune the Sentaurus Workbench schedule settings in this case.

Job Polling Occurs Too Frequently

This issue can occur on overworked or slow TM clusters. After Sentaurus Workbench submits node jobs in the TM environment with the `qsub` command, it starts a periodical check of job polling by calling the `qstat` command to obtain the status of the submitted jobs.

Workaround

You can reduce this frequency in the Sentaurus Workbench preferences and the tool database.

The preferences setting **Scheduler** > **TM Jobs** > **Job Polling Interval (sec)** specifies the time interval (in seconds) between two sequential calls of the `qstat` command (default: 1 s). Increasing this time interval might help the TM clusters that are overworked.

RTDA Scheduler

Sentaurus Workbench provides an interface to the Runtime Design Automation (RTDA) NetworkComputer. It is used to manage a computer farm or computer cluster, and is responsible for accepting, scheduling, dispatching, and managing the remote execution of standalone, parallel, or interactive user jobs. It also manages the allocation of resources such as processors, memory, disk space, software licenses, and custom objects to jobs that require them. RTDA uses a unique event-driven scheduler that results in very low overhead per job, usually in the millisecond range.

To use RTDA, the following conditions must be fulfilled:

1. Correct RTDA environment settings are set up. Check with your systems administrator about which RTDA configuration script to source or how to modify your environment settings manually. In particular, the RTDA binaries are accessible through the environment variable `PATH`.
2. The local host (the one running Sentaurus Workbench and `gsub`) is an RTDA submit or admit host.
3. Synopsys TCAD simulation software is accessible to all RTDA hosts.
4. The project space is accessible to all RTDA hosts with the same absolute path.

To confirm points 1 and 2, the following command should succeed on the local host:

```
% vovarch
```

If this command fails, check with your systems administrator.

The `gsub` utility uses the `nc run` command to submit jobs to RTDA, the `nc list` command to track the job status, and the `nc stop` command to terminate jobs. In addition, the `nc forget` command is called for completed jobs to ensure they no longer appear in the list of jobs returned by the `nc list` command.

You can specify additional `nc run` command-line options using the queue options (see RTDA Queues on page 278). To execute each node, `gsub` does not directly submit the corresponding tool command, but launches a job wrapper, the `gjob` utility. The latter updates the job status and locally executes the prologue tool and the epilogue of the corresponding tool as defined in the tool database.

You can specify RTDA resource descriptions on a per-tool basis. Different resource descriptions can be set up for specific tools in a Sentaurus Workbench project. For descriptions of the RTDA resource flags, refer to the definition of the `-r` option of the `nc run` command in the NetworkComputer documentation.

The per-tool resource descriptions can be set up in the Sentaurus Workbench tool database (on the global, user, or project level) using the syntax:

```
set WB_tool(<tool_name>,RTDA,resource) {<resource_description>}
```

where <tool_name> is the database name of the tool, and <resource_description> defines the space list of resources preceded by the `-r+` and `-r-` options. For example:

```
set WB_tool(sprocess,RTDA,resource) {-r+ CPUS/4 -r- linux64 -r+ RAM/500}
```

The resource description, specified in the tool database, is added to the one specified in the RTDA queue in the Sentaurus Workbench global or site queue configuration files (`gqueues.dat`).

Troubleshooting RTDA

Systems administrators of overworked RTDA clusters might receive complaints from Sentaurus Workbench users about job polling occurring too frequently. This section describes how to fine-tune Sentaurus Workbench schedule settings in this case.

Job Polling Occurs Too Frequently

This issue can occur on overworked or slow RTDA clusters. After Sentaurus Workbench submits node jobs in the RTDA environment with the `nc run` command, it starts a periodical check of job polling by calling the `nc list` command to obtain the status of the submitted jobs.

Workaround

You can reduce this frequency in the Sentaurus Workbench preferences and the tool database.

The preferences setting **Scheduler** > **RTDA Jobs** > **Job Polling Interval (sec)** specifies the time interval (in seconds) between two sequential calls of the `nc list` command (default: 1 s). Increasing this time interval might help RTDA clusters that are overworked.

Configuring Scheduling Systems

Scheduling systems require a queue configuration file `gqueues.dat` to be defined on a global or site level. The systems administrator must set up this file.

In addition to the queue configuration, tools must be properly associated with the queues in scheduling systems.

Global Queue Configuration File

A queue configuration file (`gqueues.dat`) combines definitions of all resources available to you for running simulations in Sentaurus Workbench. The file contains a list of Sentaurus Workbench queue definitions.

Each Sentaurus Workbench queue consists of the name of the scheduler, a unique queue name, and optional parameters. Sentaurus Workbench and the command-line job submission utilities (`gsub` and `genopt`) use these queues when launching simulations.

From the Sentaurus Workbench side, in the global queue configuration, the systems administrator lists all the queues that you can access in your corporate computation environment. The systems administrator must also set up the resource strings, specifying applicable resource restrictions.

In the `gqueues.dat` file, lines starting with a hash (#) character are considered comments and are ignored. An example of a queue definition file is:

```
#local queues
queue local:default "19"
queue local:priority "10"

#lsf queues
queue lsf:normal "bsub"
queue lsf:mylsfqueue "bub -R \"swp > 5 && mem > 10\" -q normal"
queue lsf:sp "bsub -R \"rusage[mem=410] span[hosts=1]\" -n 4 -P  tcad
   -q normal"

#sge queues
queue sge:normal "qsub"
queue sge:normal64 "qsub -V -cwd -now n -P bnormal -l arch=glinux"
queue sge:mt4 "qsub -V -cwd -now n -P bnormal -pe mt 4"

#tm queues
queue tm:normal "qsub"
queue tm:mt4 "qsub -l nodes=1:ppn=4"

#rtda queues
queue rtda:normal "nc run -C swb"
queue rtda:mt4 "nc run -r+ CPUS/4 -r+ RAM/500 -r+ linux64"
```

Sentaurus Workbench looks for the global queue file in the following order:

1. Release-independent directory: `$STROOT/queues/gqueues.dat`
2. Release-specific directory: `$STROOT/tcad/$STRELEASE/lib/gqueues.dat`
3. Release-specific directory: `$STROOT/tcad/$STRELEASE/lib/glib2/gqueues.dat`

The global queue file also can be specified on a site level: `$SWB_SITE_SETTINGS_DIR/gqueues.dat`. When this file exists, it replaces the global one (see Site Queue Configuration on page 278).

Sentaurus Workbench supports different schedulers, which can be used simultaneously: local, LSF, SGE, TM, and RTDA. The global queue file can combine queues for all available resources. As a minimum requirement, the global queue file must contain at least one local queue to launch Sentaurus Workbench jobs locally.

Local Queues

The queue configuration for local queues (that is, for jobs that are run on a local machine) is set up as follows:

```
queue local:<queue_name> <desired nice level>
```

For example, a priority queue can have a lower nice level:

```
queue local:priority "10"
```

Sentaurus Workbench provides the following predefined local queues in the `$STROOT/tcad/$STRELEASE/lib/glib2/gqueues.dat` file:

```
queue local:default "19"
queue local:priority "0"
```

The local machine is the one where you launch Sentaurus Workbench. However, as soon as a project is launched, the local machine for that project is fixed to the host where it runs, until project execution is finished. In other words, Sentaurus Workbench detects which machine is the *local* one when execution of a project starts, and you cannot change the machine for that project until its execution is completed. The following example illustrates this concept:

1. You start Sentaurus Workbench on host `host1` and launch node `12` in a local queue. Project execution starts on host `host1`.
2. Then, you start another Sentaurus Workbench session on host `host2` and open the same project that is still running.
3. You launch node `24` in a local queue. This job starts on `host1` rather than `host2`.

LSF Queues

For installing and setting up LSF queues, refer to the LSF installation manual.

The format of the LSF queue specification string for the global queue configuration file is:

```
queue lsf:<lsf_queue_name> "bsub <constraint_options>"
```

where:

- `<lsf_queue_name>` is the name of the corresponding LSF queue.
- `<constraint_options>` contains arbitrary constraint specifications and the command-line options of the `bsub` command.

For example, a queue `<mylsfqueue>` with additional resource constraints can be defined for Sentaurus Workbench as:

```
queue lsf:mylsfqueue "bsub -R \"linux && mem>1000\""
```

When Sentaurus Workbench runs a job using this queue, the following command will be generated on the basis of the queue definition:

```
bsub -R "linux && mem>1000" -o <log_file> -q mylsfqueue \
   -P <project_id> -J <job_name> <gjob_command_string>
```

Note:

You must use the backslash (\) in the specification of an LSF resource string to protect double quotation marks. Otherwise, Sentaurus Workbench removes double quotation marks during Tcl evaluation and the LSF `bsub` command fails on the incorrect syntax.

The name of a real LSF queue and the name of the corresponding queue in the global or site queue configuration file of Sentaurus Workbench usually are the same, but it is not mandatory. Sentaurus Workbench allows you to create *queue aliases*, where Sentaurus Workbench queues have different names and resource strings, and refer to the same LSF queue. For example:

```
queue lsf:refqueue1  "bsub -R \"swp > 5 && mem > 10\" -q normal"
queue lsf:refqueue2  "bsub -R \"swp > 10 && mem > 20\" -q normal"
queue lsf:refqueue3  "bsub -R \"swp > 15 && mem > 30\" -q normal"
```

These Sentaurus Workbench queues `refqueue1`, `refqueue2`, and `refqueue3` are not real LSF queues, that is, they do not exist in the LSF environment. In fact, these queues refer to the same LSF `normal` queue. The LSF `bsub` command supports multiple `-q` command-line options, but the last one specifies the LSF queue for the execution. That is why, in the above example, simulations scheduled among these three queues will be submitted to the same LSF queue `normal`, but with different resource restrictions.

When the `lsf:default` queue for LSF is specified in the global or site queue configuration file, Sentaurus Workbench expects explicit specification of the LSF queue in the constraint options. For example:

```
queue lsf:default "bsub -R \"linux && mem>1000\" -q mylsfqueue"
```

For additional `bsub` options, refer to the LSF documentation.

SGE Queues

For installing and setting up SGE queues, refer to the SGE documentation.

The format of the SGE queue specification string for the global or site queue configuration file is:

```
queue sge:<sge_queue_name> "qsub <arguments>"
```

where:

- `<sge_queue_name>` is the SGE queue name that you will see in Sentaurus Workbench.
- `<arguments>` contains the list of options. It can contain arbitrary resource descriptors to filter out hosts with requested resources.

The main option is `-P`, which specifies the name of the associated SGE project. The SGE project specifies the group of SGE hosts where the simulation job is to be scheduled.

For example, a queue `<mysgequeue>` with additional resource descriptors can be defined for Sentaurus Workbench as:

```
queue sge:mylsfqueue "qsub -P bnormal -l
   arch=glinux,kernel_version=2.6.9"
```

When Sentaurus Workbench runs a job using this SGE queue, it first packs the `gjob` call into the shell script:

```
#!/bin/sh
#$ -S /bin/sh
trap "kill -INT $!;sleep(5);kill -TERM $$" USR1 USR2
<gjob command string>
```

and then submits this script to the SGE engine using the command:

```
qsub -notify -V -cwd -now n -P bnormal -l
arch=glinux,kernel_version=2.6.9 -o <log_file>
-e <sge_error_file> -N <project_name> <shell script above>
```

This command returns the job ID, which Sentaurus Workbench uses to monitor the progress of the job by periodically calling the command:

```
qstat -u <user_name> -j <jobID from qsub>
```

For additional `qsub` and `qstat` options, refer to the SGE documentation.

TM Queues

For installing and setting up TM queues, refer to the TORQUE Resource Manager documentation.

The syntax of TM queues is similar to that for SGE queues since both schedulers are based on the same PBS scheduling system:

```
queue tm:<sge_queue_name> "qsub <arguments>"
```

Note that `qsub` arguments might differ in SGE and TM. For additional `qsub` and `qstat` options, refer to the TORQUE Resource Manager documentation.

RTDA Queues

For installing and setting up RTDA queues and job classes, refer to the NetworkComputer documentation.

The format of the RTDA queue specification string for the global queue configuration file is:

```
queue rtda:<rtda_queue_name> "nc run <options>"
```

where `<rtda_queue_name>` is the name of the corresponding RTDA queue, and `<options>` contains arbitrary constraint specifications and instructions for the `nc run` command.

For example, a queue `<mt4>` with additional resource constraints can be defined for Sentaurus Workbench as:

```
queue rtda:mt4 "nc run -C swb -r+ CPUS/4 -r+ RAM/500 -r+ linux64"
```

When Sentaurus Workbench runs a job using this queue, the following command is generated on the basis of the queue definition:

```
nc run -C nc_swb0 -r+ CPUS/4 -r+ RAM/500 -r+ linux64 -l <log_file> \
    -J <job_name> -v 1 -rundir <project_dir> <log_file>
```

For additional `nc run` options, refer to the NetworkComputer documentation.

Site Queue Configuration

When your company has multiple groups of users of the Sentaurus platform, distributed in different sites, it might be useful to redefine global queue configuration settings for all users of the specific site.

To do this:

1. Create your site-specific settings in the `gqueues.dat` file.
2. Place the file under an arbitrary directory.
3. Set up the following environment variable to refer to this directory:

   ```
   % setenv SWB_SITE_SETTINGS_DIR <path_to_site_directory>
   ```

Like the global queue configuration file, its site counterpart is usually not writable.

The directory referenced by the environment variable `SWB_SITE_SETTINGS_DIR` can also contain the site tool database settings file `tooldb.tcl`. Unlike the site tool database file, the site queue configuration file completely replaces the settings provided by its global counterpart. Therefore, when a site-specific `gqueues.dat` file is specified, the global `gqueues.dat` file is not mandatory.

Note:

The site-level queue configuration file allows you to create your own queue settings that differ from those at the global level.

Tool Associations

Sentaurus Workbench supports the following levels of tool-queue associations: global, user, project, and node. The systems administrator sets up the global tool-queue association. You can configure the other three levels to meet your requirements.

Note:

You cannot use the `queue` keyword in user-project queues files. This means you cannot set up new queues on the user and project levels. These levels have permission to assign the tools and the nodes to specific queues.

Global, user, and project tool-queue associations have a specific format:

```
tool <tool_name> "options" <scheduler:queue_name>
```

The following example defines the running of all nodes associated with the Sentaurus Process tool on the hosts specified by the `lsf:normal` queue:

```
tool sprocess "" lsf:normal
```

You can use the keyword `default` to specify all tools. For example, the following assignment means that all tools will run in the `sge:mt4` queue with the subset of hosts provided in the options field:

```
tool default "" sge:mt4
```

Options are specific to the scheduler used and differ accordingly (see Global Queue Configuration File on page 274).

Note:

Of the following options, the user tool associations and project tool associations can be modified on the **Scheduler** tab. The node-specific constraints can be edited manually from the **Scheduler** tab or by using a text editor (see Modifying Project-Level Tool Queues on page 84).

Global Tool Associations

Global tool associations are written to the global queue configuration file or site queue configuration file or both, and are loaded with the setup. These associations apply to all users.

Sentaurus Workbench delivers the following default tool associations in the global queue file `$STROOT/tcad/$STRELEASE/lib/glib2/gqueues.dat`:

```
tool inspect "" local:default
tool svisual "" local:default
tool bridge  "" local:default
tool default "" local:default
```

These tool associations are used to run interactive tools locally regardless of the queue to which they are scheduled.

User Tool Associations

You can use the list of queues and prepare the assignment of tools to queues. Options can be modified according to the requirements. The user tool associations are stored in the file `gqueues_<username>.dat` in the `$STDB` directory of the user. All the definitions specified complement and override any definitions at the global level and apply to all of your projects.

Project Tool Associations

You can configure the tool queue associations at the project level. These definitions are stored in the `gqueues.dat` file in the project directory.

All definitions override any setting at the global level or user level, and apply to the current project only.

For the local scheduler, the option field represents the nice level at which the tool must be run. For other schedulers, the option field redefines the resource string.

Node-Specific Constraints

Node-specific constraints are handled by expanding the expression and applying the constraints to the nodes that match the expression. Node-level constraints are applied during runtime:

```
node "gexpr" <queue:name> "options"
```

For example:

```
node "all:{ @P1@ > 2.0 }" local:default "11"
```

Node-specific constraints can be set up at the global, user, or project level. These settings are stored in the respective definition files as previously mentioned.

Note:

Definitions are cascaded with the latest setting overwriting the setting of the previous level, that is, the global and site queue files are loaded initially, and the global or site tool-queue associations and their corresponding strings are loaded. Then, the user-queue definition file is loaded. Any previous definition related to a particular tool is overwritten. Similarly, the project queue definition overwrites the user definitions.

You can set up queues on the global and site levels only. However, tool queue assignments can be overwritten on other levels.

Extended Scheduler Log

The Sentaurus Workbench Scheduler can generate detailed log information, which can help to debug scheduling-related problems. This additional information is accumulated in the project log file (`glog.txt`) and in the node-related job files (`n<nkey>_<acr>.job`).

The `GSUB_ADVANCED_LOG` environment variable controls the volume of data logged. To switch on the advanced log feature, specify:

```
GSUB_ADVANCED_LOG 1
```

By default, this feature is switched off (`0` or undefined).

Note:

The advanced log feature must be used cautiously, since it adds large amounts of data to project-related files.

Launching Sentaurus Workbench as an Interactive Job on the Cluster

Some clusters offer the possibility to launch interactive jobs. For example, the LSF and SGE schedulers can launch an interactive terminal job on the cluster, which can be used to work with Sentaurus Workbench. Alternatively, clusters allow direct launching of dedicated interactive jobs, in which case, a so-called pseudo-terminal job starts automatically.

To launch an interactive terminal job on LSF, use the following command:

```
bsub -q <iqueue> -I xterm
```

where `<iqueue>` is the name of the LSF queue on your cluster that supports interactive jobs.

To launch an interactive terminal job on SGE, use the following command:

```
qsh -cwd -P <iproject>
```

where `<iproject>` is the name of the SGE project on your cluster that supports interactive jobs.

Note:

Ensure that you have either correctly set up your `DISPLAY` environment variable or specified a correct X display name on the command line of the submission command.

The actual submission commands might look different. Check with your corporate IT team for details and usage policy.

Running user interface applications as interactive jobs on compute clusters has become more common because many companies want to reduce the number of dedicated Linux machines. To benefit from this way of working with Sentaurus Workbench, you must consider the CPU resources your interactive job requires to prevent unwanted performance slowdown.

The cluster allocates CPU resources for your interactive job (slot). You need to take into account that Sentaurus Workbench itself and every additional process you launch from Sentaurus Workbench will share the same slot that the cluster has allocated for your interactive jobs.

Batch Processes

Sentaurus Workbench is a serial process. However, the Sentaurus Workbench infrastructure includes other batch processes running in the background. Sentaurus Workbench launches these processes on the same host, that is, they run on your interactive slot and share the same resources:

- `swblm`

  This daemon process facilitates communication between all the Sentaurus Workbench processes run by one user. It is a TCAD release–specific process, so if you run different releases of Sentaurus Workbench concurrently, you will have multiple `swblm` instances on the same host.

  This daemon process typically does not consume much CPU or memory resources. However, if you run large DoE projects with fast-running simulations, `swblm` can run in parallel (up to 20 threads) and consume significant CPU resources.

- `gsub`

  For every project you run, Sentaurus Workbench launches the `gsub` process that manages project scheduling. Each `gsub` is a serial process. However, for large DoE projects, `gsub` can consume significant CPU time.

- `gjob` + simulation processes

  By default, Sentaurus Workbench redirects Sentaurus Visual and Inspect jobs to local execution regardless of which cluster queue you selected in Sentaurus Workbench for project execution, unless it is redefined in the queue file (`$STROOT/queues/gqueues.dat`). Depending on the specified maximum of the simultaneously running local jobs (see the Sentaurus Workbench preferences; default is 1), you might have one or multiple Sentaurus Visual or Inspect processes running concurrently on the same interactive slot where Sentaurus Workbench runs. In addition, for each of these processes, you will have a Sentaurus Workbench job wrapper process `gjob`. Despite `gjob` being a serial process, it can consume significant CPU time during runtime preprocessing of the project.

If you encounter performance issues when running Sentaurus Workbench as an interactive job on the cluster, do the following:

1. Check with your corporate IT team about which CPU resources are allocated for interactive jobs. By default, it is typical to have one or even fewer CPU cores per interactive job.
2. Try to allocate more CPU resources to an interactive job.

   For example, you can allocate four CPUs for a terminal job on the same host as follows:

   ```
   LSF: bsub -q <iqueue> -R "span[hosts=1]" -n 4 -I xterm
   SGE: qsh -cwd -P <iproject> -pe mt 4
   ```

The optimal CPU resources for a Sentaurus Workbench interactive job depend on the way you work with Sentaurus Workbench. If you run a few small or medium-sized projects in one Sentaurus Workbench instance, one CPU core is sufficient for a Sentaurus Workbench interactive job on the cluster. If you launch several large DoE projects in the same Sentaurus Workbench instance, the Sentaurus Workbench infrastructure would require more resources.

In the case of performance slowdown, you can try to increase the number of cores and to find reasonable CPU resources required for Sentaurus Workbench interactive jobs.

12

Organization of Projects

This chapter describes the hierarchical project organization as an alternative to the traditional project organization in Sentaurus Workbench.

Limitations of the Traditional Project Organization

Sentaurus Workbench traditionally supports the organization of projects where all files are placed inside one directory that contains the following (see Figure 78):

- Core data such as tool inputs, simulation tool flow, design-of-experiments (DoE) table, and variables. This includes other user-specific data such as preferences, queue settings, and project run limits quota.
- Reproducible data such as preprocessed inputs, results, log files, and error files.

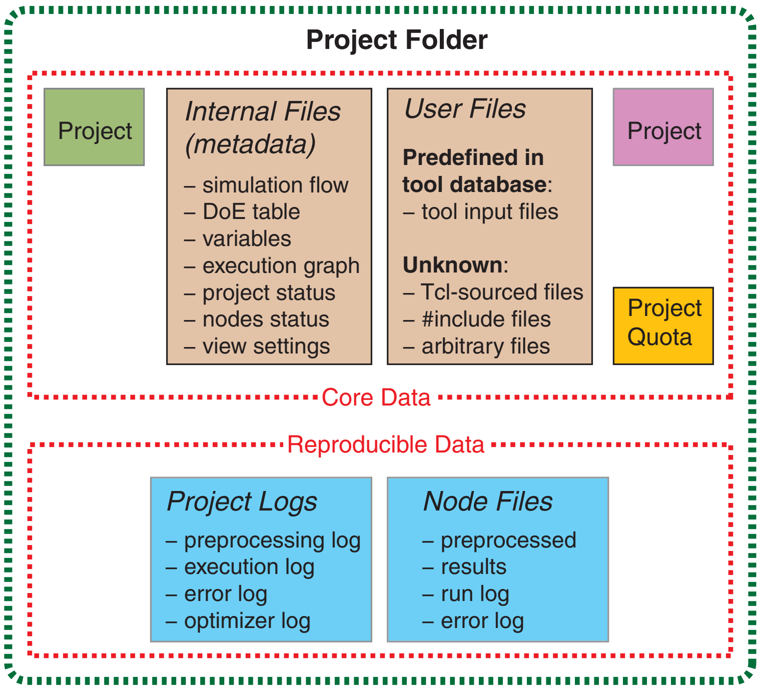
While the traditional project organization is simple and easy to use, it has limitations when working with large DoE projects. As soon as the number of experiments exceeds a certain number (usually, one hundred or more), Sentaurus Workbench slows down on multiple queries, such as:

- Deleting experiments, parametric steps, and tool instances.
- Copying and pasting selected experiments.
- Saving experiments as a new project.
- Cleaning up selected nodes and the entire project.

To identify files belonging to a given node, Sentaurus Workbench launches queries on a traditional project directory using output file patterns defined in the tool database. This is the main cause of the slowdown, because querying a directory with thousands of files takes substantial time.

With the traditional project organization, you cannot split project files to store them on different disks. In addition, Sentaurus Workbench does not allow renumbering of nodes without losing the simulation results.

Figure 78 Traditional project organization of files in Sentaurus Workbench



Hierarchical Project Organization

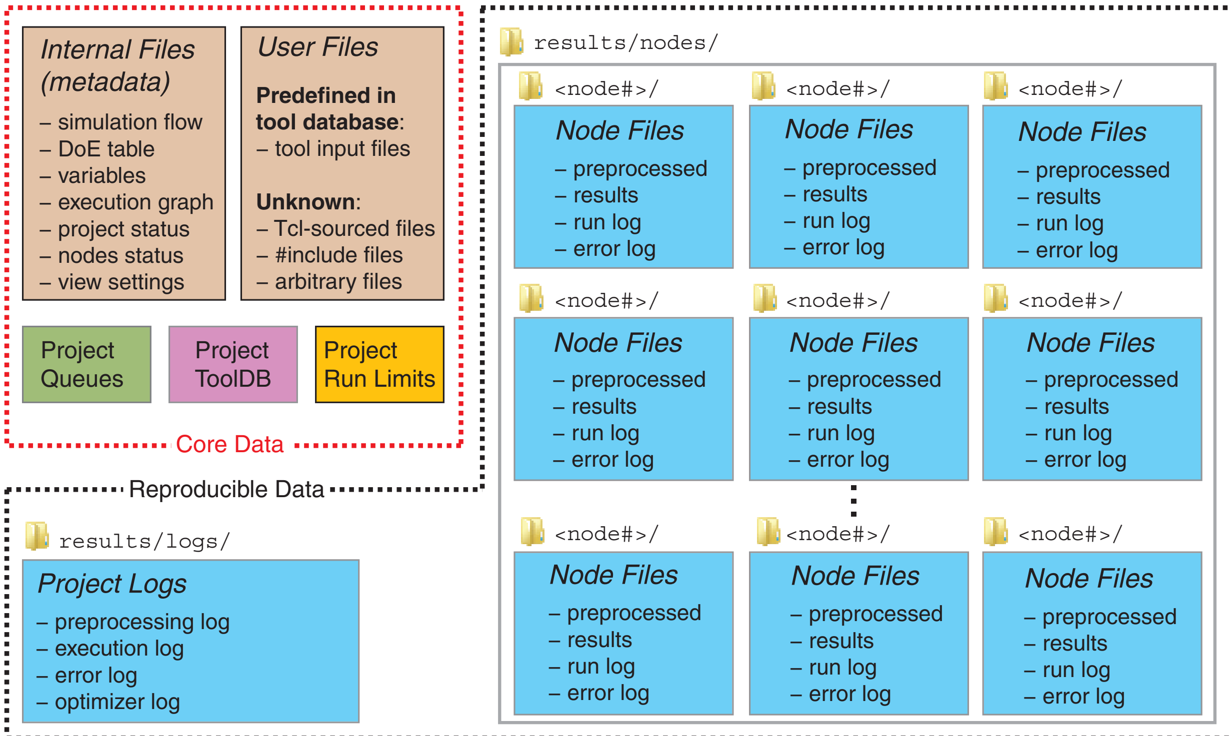
For some of your simulation projects, you can benefit from using hierarchical project organization. Sentaurus Workbench allows you to create new hierarchical projects and to convert existing traditional projects. You can work with projects in traditional and hierarchical organizations in Sentaurus Workbench.

Location of Project Files

With the hierarchical project organization (see Figure 79):

- Core data files are separated from reproducible data files. Core data files are stored under the root project directory as in the traditional project organization.
- Reproducible data files are stored in a node-wise hierarchy.

Figure 79 Hierarchical project organization of files in Sentaurus Workbench

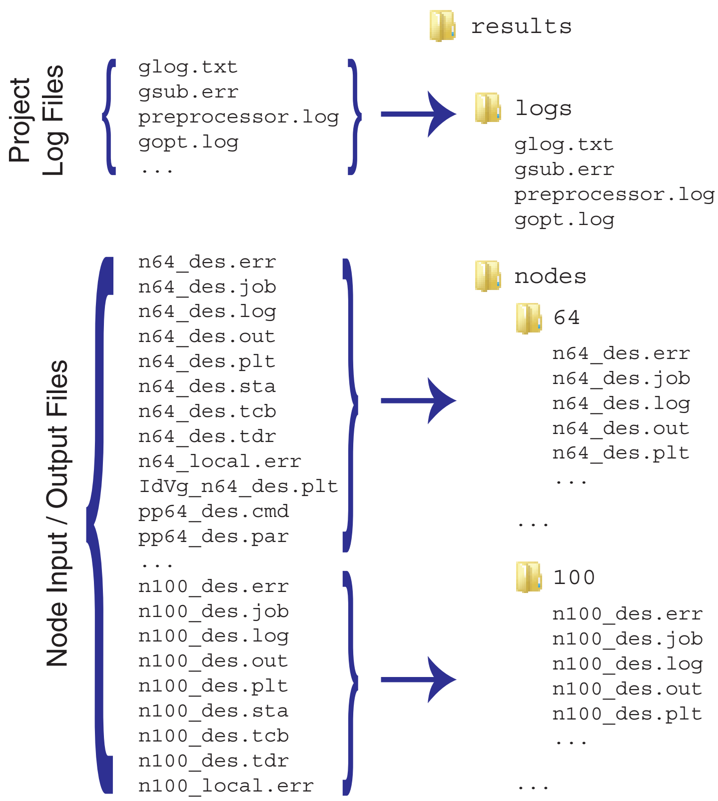


Reproducible data files, such as those generated by the execution or preprocessing of a project, are the largest part of a project. All reproducible data files are stored in the `results` subdirectory of the root project directory:

- Project log files are stored in the `results/logs` folder.
- Node files are stored in individual `results/nodes/<node#>` folders, without any further structure (see Figure 80 on page 287). The naming of node files remains the same as in the traditional project organization.

  Note:
  Virtual nodes do not have folders.

Figure 80 Example of how files are stored comparing (left) the traditional project organization with (right) the hierarchical project organization



Advantages of the Hierarchical Project Organization

This section discusses the advantages of using the hierarchical project organization.

Better Performance

The hierarchical project organization significantly improves performance with large DoE projects.

For example, deleting and cleaning up operations on a simulation project with 1300 experiments (~3000 nodes and ~80000 files) take only several seconds to complete. In contrast, the same operations take more than 20 minutes on a simulation project that uses the traditional project organization. Similarly, saving selected experiments as a new project (see Saving Projects on page 44) is much faster compared to the traditional project organization.

Split Storage of Project Files

Activating the hierarchical project organization allows you to split the storage of project files on to different disks (see Separate Storage of Project Files).

Renumbering Nodes Without Cleaning Up a Project

With the traditional project organization, you can renumber nodes only when cleaning up a projects. With the hierarchical project organization, you can renumber nodes without losing the simulation results (see Renumbering Nodes Without Cleaning Up a Project on page 128).

Separate Storage of Project Files

Sentaurus Workbench projects usually require substantial storage space due to the size of TDR files, even though TDR files are compressed internally.

With the hierarchical project organization switched on, you can store the core and reproducible data files of projects separately, thereby optimizing the use of disk space. For example, you might want to store core data files on the backup disk and reproducible data files on the scratch disk.

By default, core data files and reproducible data files are stored together (see Figure 79 on page 286). In other words, the `results` subdirectory containing project log files and node files is stored under the root project directory.

To change the location of the reproducible data files:

1. Choose **Edit** > **Preferences**, or press the F12 key.

   The SWB Preferences dialog box opens.

2. Expand **Project** > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location**.

3. Specify the path where you want to store the output files of projects.

   The default is `@STDB@`, indicating that core data files and reproducible data files are stored together.

4. Click **Apply**.

All projects under the current `STDB` directory share the same output storage rules (see Figure 81 on page 289). Each preprocessed or executed project under the current `STDB` directory stores its reproducible data files in the same hierarchical structure.

Practically, Sentaurus Workbench works with two separate hierarchies in the file system:

- The core files hierarchy (`STDB`) contains only core data files.
- The output files hierarchy (from preferences, expand **Project** > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location**) contains only reproducible data files.

Both the core files and output files hierarchies have a symmetric structure related to their root directories, that is, a project has the same relative path in both hierarchies. For example, if the core data files of a project are stored in the following folder:

```
$STDB/the/path/to/myTestProject
```

the reproducible data files of this project are stored in the following folder:

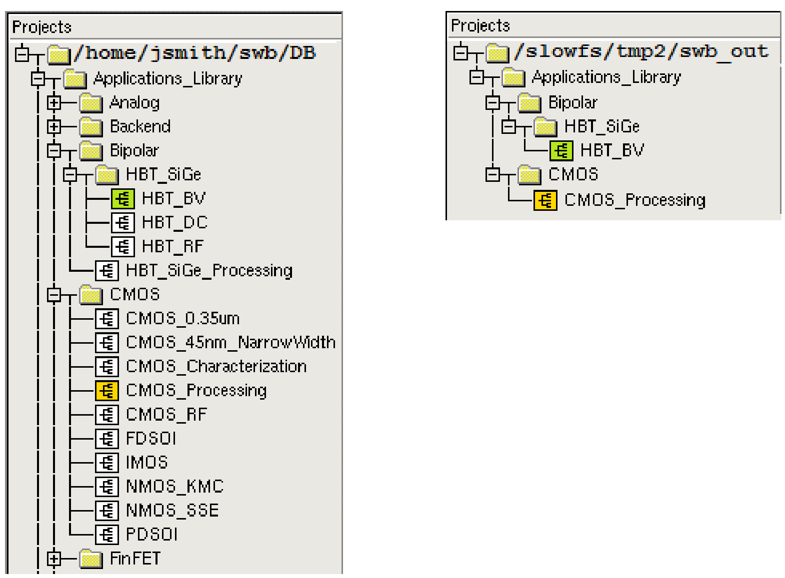
```
[Project > Organization > Settings for Hierarchical Project
Organization > Project Output Files Location]/the/path/to/myTestProject
```

The core files hierarchy referenced by `STDB` is the one that Sentaurus Workbench displays in the projects browser (Projects panel). The output files hierarchy is used only to store generated files and is not shown in Sentaurus Workbench.

Figure 81 Example of the hierarchy used for the separate storage of (left) core files and (right) output files

`STDB= /home/jsmith/swb/DB`

Project > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location**=
`/slowfs/tmp2/swb_out`



Sentaurus Workbench supports projects stored outside the `STDB` directory, provided the required file system permissions are available. If you instructed Sentaurus Workbench to store the output files of a project separately from the core files, the path in the output files hierarchy cannot be the same as its relative path in the core files hierarchy. In this case, Sentaurus Workbench adds the absolute path of the core project directory to the root output files hierarchy (see Figure 82).

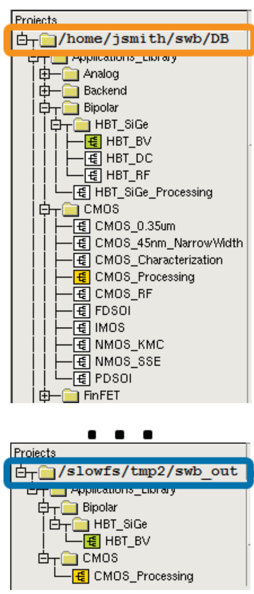
Note:

A Sentaurus Workbench project itself does not contain any information about where its output data is stored.

If you want to copy, move, delete, or export a project outside Sentaurus Workbench (for example, in a terminal window), you must take care of the files stored in the output files hierarchy.

A TCAD administrator can establish the output storage policy for Sentaurus Workbench users. That is, you can force the default location of output files to all users or propagate it to new Sentaurus Workbench users.

Figure 82 Example of rules for the location of output files



Name	Value
STDB	/home/jsmith/swb/DB
Preferences: Miscellaneous > Project Output Files Location	/slowfs/tmp2/swb_out
Use Case 1: Project inside STDB	
Path to core files:	/home/jsmith/swb/DB/cust/CMOS/test/iter2
Path to output files:	/slowfs/tmp2/swb_out/cust/CMOS/test/iter2
Use Case 2: Project outside STDB	
Path to core files:	/remote/dist/app/MyCMOS
Path to output files:	/slowfs/tmp2/swb_out/remote/dist/app/MyCMOS

The path to the location of output files can include environment variables enclosed in `@...@`. Sentaurus Workbench substitutes these settings with the real values of the corresponding environment variables to detect the target path.

For example, a TCAD administrator can set up the following user-specific path in a convenient file system to store reproducible data files:

In preferences:
Project > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location** = `/sim/tmp/@USER@/tcad/`

The following setting will make this path release-specific:

In preferences:
Project > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location** = `/sim/tmp/@USER@/tcad/@STRELEASE@`

If you work with multiple `STDB` directories, there might be a conflict when multiple `STDB` directories have the same project in the same place. To prevent this, include `STDB` in the path. For example:

In preferences:
Project > **Organization** > **Settings for Hierarchical Project Organization** > **Project Output Files Location** = `/sim/tmp/@USER@/tcad/@STDB@`

Migration to the Hierarchical Project Organization

The following rules will help you to migrate from the traditional project organization to the hierarchical project organization:

- Remember the following:
  - Node files (preprocessed and results) are stored in individual node folders.
  - Each simulation is executed in a node folder, rather than in a root project directory.
- Stop assuming that all project files are stored under the root project directory.
- If necessary, apply changes to old projects, that is, adjust tool input files, so that the projects work using the hierarchical project organization.

Example: Sentaurus Visual input file assumes all node files are in the same directory

With the traditional project organization, you would use:

```
load_file IdVg_n@node|sdevice@_des.plt -name PLT($N)
```

With the hierarchical project organization, you would use:

```
load_file ../@node|sdevice@/IdVg_n@node|sdevice@_des.plt -name PLT($N)
```

The following command will work in both traditional and hierarchical organizations:

```
load_file @[relpath IdVg_n@node|sdevice@_des.plt]@ -name PLT($N)
```

Example: Sentaurus Process input file assumes the current directory is the root project directory

With the traditional project organization, you would use:

```
icwb filename= "SRAM_lyt.mac" scale=1e-3
```

With the hierarchical project organization, you would use:

```
icwb filename= "@pwd@/SRAM_lyt.mac" scale=1e-3
```

The command will work in both traditional and hierarchical organizations.

Extended Preprocessor Syntax

With the hierarchical project organization switched on, Sentaurus Workbench launches simulation tasks in the dedicated node folders. The preprocessor substitutes `@...@` references to node files, with their corresponding relative path to the current node folder.

In addition, Sentaurus Workbench provides syntax to allow you to set up your tool input command files so that they run properly regardless of the project organization (see Table 23).

Table 23 Preprocessor references and their substitution

Preprocessor syntax	Description	Traditional project organization	Hierarchical project organization
`@node@`	Node number	`8`	`8`
`@node|sdevice@`	Node of the last Sentaurus Device tool instance	`6`	`6`
`@tdr@`	TDR file – output of process simulation and structure generation tools, such as Sentaurus Process, Sentaurus Mesh, and Sentaurus Structure Editor	`n8_fps.tdr`	`../8/n8_fps.tdr`
`@tdr|-1@`	TDR file generated on the previous step	`n7_fps.tdr`	`../7/n7_fps.tdr`
`@tdrdat@`	TDR file – output of device simulation tools, such as Sentaurus Device and Sentaurus Device QTX	`n12_des.tdr`	`../12/n12_des.tdr`
`@tdrdat|BVDss@`	TDR file generated by device simulation tool named BVDss	`n10_des.tdr`	`../10/n10_des.tdr`

Table 23 Preprocessor references and their substitution (Continued)

Preprocessor syntax	Description	Traditional project organization	Hierarchical project organization
`@plot@`	Plot file	`n12_des.plt`	`../12/n12_des.plt`
`@prjorg@`	Project organization	`traditional`	`hierarchical`
`@pwd@`	Project path in core files hierarchy (`STDB`)	`$STDB/path/to/prj`	`$STDB/path/to/prj`
`@pwdout@`	Project path in output files hierarchy	`$STDB/path/to/prj`	Common storage: `$STDB/path/to/prj` Separate storage: `/out_root/path/to/prj`
`@nodedir@`	Path to node folder, relative to `@pwdout@`	`.`	`results/nodes/8`
`@nodedirpath@`	Full path to node folder	`$STDB/path/to/prj`	Common storage: `$STDB/path/to/prj/results/nodes/8` Separate storage: `/out_root/path/to/prj/results/nodes/8`
`@nodesdir@`	Full path to directory with node folders	`$STDB/path/to/prj`	Common storage: `$STDB/path/to/prj/results/nodes` Separate storage: `/out_root/path/to/prj/results/nodes`
`@logsdir@`	Full path to directory with project log files	`$STDB/path/to/prj`	Common storage: `$STDB/path/to/prj/results/logs` Separate storage: `/out_root/path/to/prj/results/logs`
`@[relpath FILE]@`	Path to a file relative to the current node folder	`FILE`	`../8/FILE`
`@[abspath FILE]@`	Full path to a file	`$STDB/path/to/prj/FILE`	Common storage: `$STDB/path/to/prj/results/nodes/8/FILE` Separate storage: `/out_root/path/to/prj/results/nodes/8/FILE`

Sentaurus Workbench supports special preprocessor commands to ease the creation of tool input command files, which work in both project organizations. When you refer to a file that does not belong to the current node, use one of these commands:

```
@[relpath <Reference_to_node_file>]@
```

```
@[abspath <Reference_to_node_file>]@
```

These commands instruct Sentaurus Workbench that the reference points to a file that belongs to another node rather than the current one. Sentaurus Workbench correctly resolves these references during preprocessing and execution. This allows you to set up tool input command files without thinking about project organization. For example, the following line in a Sentaurus Visual command file:

```
load_file @[relpath IdVg_n@node|sdevice@_des.plt]@ -name
PLT(@node|sdevice@)
```

is replaced with the following line in a traditional project:

```
load_file IdVg_n45_des.plt -name PLT(45)
```

and with the following line in a hierarchical project:

```
load_file ../45/IdVg_n45_des.plt -name PLT(45)
```

Now, a similar instruction with the `abspath` command will be resolved with absolute paths:

```
load_file @[abspath IdVg_n@node|sdevice@_des.plt]@
   -name PLT(@node|sdevice@)
```

and is replaced with the following line in a traditional project:

```
load_file /path/to/prj/IdVg_n45_des.plt -name PLT(45)
```

and with the following line in a hierarchical project:

```
load_file /path/to/prj/results/nodes/45/IdVg_n45_des.plt -name PLT(45)
```

Note that `@...@` file references are substituted differently inside and outside of the `@[relpath ...]@` and `@[abspath ...]@` commands in hierarchical projects. For example, `XYZ_@plot@` is substituted as `XYZ../123/n123_des.plt`. However, the expression `@[relpath XYZ_@plot@]@` is substituted as `../123/XYZ_n123_des.plt`.

When you need a logical split between traditional and hierarchical organizations in a tool input command file, use the following conditional preprocessor statement:

```
#if "@prjorg@" == "hierarchical"
   # commands for hierarchical mode
   ...
#else
   # commands for traditional mode
   ...
#endif
```

A

Preprocessor and Reference Syntax

This appendix discusses the syntax of @-references and preprocessor commands.

@-References and Tree Navigation

The syntax of @-references in EBNF notation is:

```
reference: simple_reference [ operator [operator] ]

simple_reference: ("node" | "previous" | file_type ["/i"] | ["/o"] |
    "experiment" | "experiments" | "process_name" | "swb_parameter" |
    parameter_name | variable_name)

operator: (":"|"|") (["+"|"-"] number | tool_label | "first" |
    "last" | "index" | "all" | "min" | "max")
```

where:

`file_type`	One of the file types defined in the tool database.
`/i` and `/o`	Extensions to `file_type` refer to the corresponding input and output files of the current tool. Without extensions, an implicit input file generated by a preceding tool is searched further up the tree.
`parameter_name`	One of the declared parameters.
`variable_name`	One of the known variables.
`tool_label`	The label of a tool instance in the simulation flow. The corresponding tool is defined in the tool database of Sentaurus Workbench.
`experiment`	Returns the first experiment to which a node belongs.
`experiments`	Returns all experiments to which a node belongs.

`process_name`	Returns the process name to which the current node belongs.
`swb_parameter`	Returns the parameter name of the current node.

The following relative direction suffixes can be used (depending on the flow orientation):

`|`	Horizontal flow orientation: horizontal navigation operator. Vertical flow orientation: vertical navigation operator.
`:`	Horizontal flow orientation: vertical navigation operator. Vertical flow orientation: horizontal navigation operator.
`+number`	Horizontal flow orientation: relative reference in right horizontal or downward vertical direction. Vertical flow orientation: relative reference in downward vertical or right horizontal direction.
`-number`	Horizontal flow orientation: relative reference in left horizontal or upward vertical direction. Vertical flow orientation: relative reference in upward vertical or left horizontal direction.
`number`	Horizontal flow orientation: absolute index reference in right horizontal or downward vertical direction. Vertical flow orientation: absolute index reference in downward vertical or right horizontal direction.

Vertical and horizontal navigation operators can be combined.

Horizontal Flow Orientation

In the vertical direction, a unit represents an entire simulation phase. Intermediate nodes (virtual nodes or split points) are not taken into account. For horizontal operators, the following additional keywords can be used instead of numbers:

`"all"`	Returns a list of all references at the indicated horizontal level.
`"first"`	Returns the first, leftmost reference at the indicated level.
`"last"`	Returns the last, rightmost reference at the indicated level.
`"index"`	Returns the horizontal index of the node at the respective level instead of the node number, with 1 being the index of the leftmost node.

`"min"`	Returns the index of the leftmost node at the respective level (always 1).
`"max"`	Returns the index of the rightmost node at the respective level.

With a vertical operator, you also can use the tool instance label (`tool_label`) as an absolute position indicator.

The reference evaluates to:

- A horizontal node index if `index`, `min`, or `max` is used as the horizontal operator.
- A parameter value if `parameter_name` is used as the reference.
- A file name if `file_type` is used as the reference.
- A tool label if `tool_label` is used as the reference.
- A node key if `node` or `previous` is used as the reference.

If several values result from the reference, a space-separated list is returned. Examples are:

`@node@`	Current node key (the key of the output node of the current tool instance).
`@node:all@`	List of all node keys at the current tree level, that is, at the level of the current tool output nodes.
`@node:2@`	Node key of the second node at the current tree level.
`@node:+2@`	Node key two positions below the current node (in the same column).
`@node:-1@`	Node key immediately above the current node (in the same column).
`@node:first@`	Uppermost node key at the current tree level (column).
`@node|-1:all@`	All nodes on the previous level (column).
`@node|+3@`	Node keys at the third generation of siblings of the current node (virtual nodes are ignored), three real levels to the right.
`@node:index@`	Index of the current node.
`@node:min@`	First index at the current tree level (always 1).
`@file_type@`	Output file of the first preceding matching tool.
`@file_type:all@`	List of all file names of type `file_type` produced by the tool at the current tree level.

`@file_type:3@`	File name of type `file_type` of the third tool instance at the current tree level.
`@file_type:+1@`	File of type `file_type` at the node below the current node (in the same column).
`@file_type:last@`	Rightmost file of type `file_type` at the current level.
`@file_type/i@`	Input file of type `file_type` of the current tool (for example, `n5_mesh.tdr`).
`@file_type/o@`	Output file of type `file_type` of the current tool if defined as an output file.
`@file_type/o|-1@`	Output file of type `file_type` of the previous tool if defined as an output file.
`@node:max@`	Last index at the current tree level, the number of nodes at the current level.
`@tool_label@`	Label of the tool at the current node.
`@tool_label|all@`	List of labels of tools appearing in the simulation flow.
`@tool_label|1@`	Label of the first tool in the simulation flow, the tool at absolute level 1.
`@tool_label|+1@`	Label of the following tool.
`@tool_label|-1@`	Label of the previous tool.
`@tool_label|first@`	Label of the first tool in the simulation flow.
`@tool_label|last@`	Label of the last tool in the simulation flow.

Vertical Flow Orientation

In the horizontal direction, a unit represents an entire simulation phase. Intermediate nodes (virtual nodes or split points) are not taken into account. For vertical operators, the following additional keywords can be used instead of numbers:

`"all"`	Returns a list of all references at the indicated vertical level.
`"first"`	Returns the first, upward reference at the indicated level.
`"last"`	Returns the last, downward reference at the indicated level.

`"index"`	Returns the vertical index of the node at the respective level instead of the node number, with 1 being the index of the upward node.
`"min"`	Returns the index of the upward node at the respective level (always 1).
`"max"`	Returns the index of the downward node at the respective level.

With a horizontal operator, you also can use the tool instance label (`tool_label`) as an absolute position indicator.

The reference evaluates to:

- A vertical node index if `index`, `min`, or `max` is used as the vertical operator.
- A parameter value if `parameter_name` is used as the reference.
- A file name if `file_type` is used as the reference.
- A tool label if `tool_label` is used as the reference.
- A node key if `node` or `previous` is used as the reference.

If several values result from the reference, a space-separated list is returned. Examples are:

`@node@`	Current node key (the key of the output node of the current tool instance).
`@node:all@`	List of all node keys at the current tree level, that is, at the level of the current tool output nodes.
`@node:2@`	Node key of the second node at the current tree level.
`@node:+2@`	Node key two positions to the right of the current node (in the same row).
`@node:-1@`	Node key immediately to the left of the current node (in the same row).
`@node:first@`	Leftmost node key at the current tree level (row).
`@node|-1:all@`	All nodes on the previous level (row).
`@node|+3@`	Node keys at the third generation of siblings of the current node (virtual nodes are ignored), three real levels to the bottom.
`@node:index@`	Index of the current node.
`@node:min@`	First index at the current tree level (always 1).
`@file_type@`	Output file of the first preceding matching tool.

`@file_type:all@`	List of all file names of type `file_type` produced by the tool at the current tree level.
`@file_type:3@`	File name of type `file_type` of the third tool instance at the current tree level.
`@file_type:+1@`	File of type `file_type` at the node to the right of the current node (in the same row).
`@file_type:last@`	Downward file of type `file_type` at the current level.
`@file_type/i@`	Input file of type `file_type` of the current tool (for example, `n5_mesh.tdr`).
`@file_type/o@`	Output file of type `file_type` of the current tool if defined as an output file.
`@file_type/o|-1@`	Output file of type `file_type` of the previous tool if defined as an output file.
`@node:max@`	Last index at the current tree level, the number of nodes at the current level.
`@tool_label@`	Label of the tool at the current node.
`@tool_label|all@`	List of labels of tools appearing in the simulation flow.
`@tool_label|1@`	Label of the first tool in the simulation flow, the tool at absolute level 1.
`@tool_label|+1@`	Label of the following tool.
`@tool_label|-1@`	Label of the previous tool.
`@tool_label|first@`	Label of the first tool in the simulation flow.
`@tool_label|last@`	Label of the last tool in the simulation flow.

You can use additional references that returns the current directory without suffixes only:

`@pwd@`	Absolute path of project.
`@pwd@/@file_type@`	File reference with absolute path.

#-Commands

Any preprocessor command starts with a hash (#) as the first character on a line. Space or tab characters are allowed after the initial # for indentation. The `spp` utility recognizes the following commands:

`#<string>`	This is a comment. `spp` strips all comment lines from the preprocessed file and replaces them with empty lines. `<string>` is any string not listed here as a preprocessor command.
`#define <name> <string>`	Defines a new macro `<name>`. `spp` replaces subsequent instances of `<name>` with `<string>`.
`#undef <name>`	Undefines a previously defined macro.
`#setdep <list of nodes>`	Explicitly sets dependencies of these nodes.
`#remdep <list of nodes>`	Explicitly removes dependencies from these nodes.
`#include "<filename>"`	Includes the contents of file name at this point. `spp` processes the included file as if it were part of the current file.
`#includeext "<filename>"`	Same as the `#include` command, but with advanced processing of `<filename>`. It allows `<filename>` to contain macros already defined with the `#define` command as well as to define new macros.
`#if <expression>`	Subsequent lines up to the matching `#else`, `#elif`, or `#endif` commands appear in the output only if `<expression>` evaluates to nonzero. `<expression>` is any standard Tcl expression that evaluates to a number. Before evaluation, all @-substitutions are expanded in `<expression>`.
`#if in <gexpr>`	Same as the previous `#if` command, but evaluated to true when the current node belongs to the nodes returned by `gexpr` (see Node Expressions on page 168).
`#ifdef <name>`	Subsequent lines up to the matching `#else` or `#endif` commands appear in the output only if a macro `<name>` has been defined previously with the `#define` command.
`#ifndef <name>`	Subsequent lines up to the matching `#else` or `#endif` commands appear in the output only if a macro `<name>` has NOT been defined previously with the `#define` command.

`#elif <expression>`	Any number of `#elif` commands can appear between an `#if` command and a matching `#else` or `#endif` command. If the expression evaluates to nonzero, subsequent `#elif` and `#else` commands are ignored up to the matching `#endif`. Any expression allowed in an `#if` command is allowed in an `#elif` command. The lines following the `#elif` command appear in the output only if all of the following conditions are met: • The expression in the preceding `#if` command is evaluated to zero. • The expressions in all preceding `#elif` commands are evaluated to zero. • The current expression evaluates to nonzero.
`#else`	Inverts the sense of the conditional command otherwise in effect. If the preceding conditional indicates that lines are to be included, lines between the `#else` and the matching `#endif` are ignored. If the preceding conditional indicates that lines would be ignored, subsequent lines are included in the output. Conditional commands and corresponding `#else` commands can be nested.
`#endif`	Ends an `#if`, `#ifdef`, or `#ifndef` section.
`#exit`	Stops preprocessing at this line. The following lines are stripped.
`#verbatim <string>`	A line starting with `#verbatim` is not stripped, that is, the prefix `#verbatim` is removed, but the rest of the line is not touched.
`#rem <string>`	A line starting with `#rem` is not stripped, but @-substitutions are expanded in the rest of the line.
`#noexec`	Does not execute (submit to scheduler) the current node.
`#set <varname> <value>`	Sets the value of `<varname>`. Shows the variable in the Variables Values view in Sentaurus Workbench.
`#seth <varname> <value>`	Sets the value of `<varname>`. Hides the variable in the Variables Values view in Sentaurus Workbench.

Split Commands

The following preprocessor commands handle split points:

`#header`	Marks the first line of the header section. The header section is copied in the beginning of each partial input file before the load command. Only one header can be defined.
`#endheader`	Last line of the header section, replaced by a blank line.
`#postheader`	A postheader is an input file section that is copied after the load command in each partial input file. This is useful to define simulator defaults, which must be reset after a load command. Any number of postheaders can be defined at any position in the original input file. They are appended after the load command in the order of their definition. Nesting or overlapping postheaders are not allowed.
`#endpostheader`	Last line of a postheader section, replaced by a blank line.
`#split @PNAME@`	Defines a split on parameter `PNAME`. The input file is cut at this point into (1) a partial input file from the previous split point to the current line and (2) another partial input file from the current line to the next split point (or to the end of the file). The current line is replaced with a save command at the end of the first partial input file. The second partial input file starts with the header, followed by the simulator load command, the optional postheaders and, finally, the real partial input section. `param_name` denotes the tree level where the split applies.
`#split PNAME`	Same as the `#split @PNAME@` command. Both syntaxes are equivalent. For legacy reasons, the `#split PNAME` command is provided to support old simulation projects. Although Sentaurus Workbench supports both syntaxes, you should use the `#split @PNAME@` command in your new simulation setups.

Note:

Multiple split points are valid only if they appear in the same order as the corresponding parameters in the simulation flow. If that is not the case, the preprocessor takes only the best possible match, ignoring certain split points.

A partial input file (corresponding to a split section) is considered a conventional tool input file regarding forward references. A common mistake is to introduce a split point after a parameter @-reference, when the referenced parameter appears after the split parameter in the simulation flow.

Every #-preprocessor command that should be unique (common) to all split files must be placed in the header section (`#header ... #endheader`). For example, the following three macro definitions will appear in all files produced by `#split`:

```
#header
...
#define macro1 string1
#define macro2 string2
#define macro3 string3
...
#endheader
```

Otherwise, these macro definitions will be placed in the first file only and will be unknown to the other files that were created after splitting. It can result in a preprocessing error.

Node Expressions

The EBNF syntax of a node expression `GEXPR` is:

```
gexpr    : gterm [operator gterm]
operator : "+" | "*" | "-" | "^"
gterm    : scnr["|"level][":{" filter "}"] | tool_label[":{"
           filter "}"] | node | "(" gterm ")" | "~" gterm
node     : integer
scnr     : "all" | identifier
level    : integer | "last" | tool
tool     : identifier
filter   : tcl_expr
```

where:

`scnr`	Name of a scenario
`tool_label`	Label of a tool instance
`node`	Node number
`level`	Tree level, starting from 0
`last`	Last tree level
`"+"`	The ‘or’ or ‘union’ binary operator
`"*"`	The ‘and’ or ‘intersection’ binary operator
`"-"`	The ‘difference’ binary operator
`"^"`	The ‘exclusive-or’ binary operator

`"~"`	The ‘extend-to-root’ binary operator
`filter`	Tcl expression

B

Menus and Toolbar Buttons of the User Interface

This appendix lists the menus and the toolbar buttons of the user interface of Sentaurus Workbench.

Project Menu

Table 24 Project menu commands

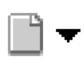
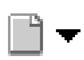
Command	Toolbar button	Shortcut keys	Description
New:			
Traditional Project		Ctrl+N	Creates a new project with traditional organization.
Hierarchical Project			Creates a new project with hierarchical organization.
Folder			Creates a new folder in the projects browser.
Open		Ctrl+O	Opens a project.
Close		Ctrl+F4	Closes currently opened project.
Save		Ctrl+S	Saves project under an existing name.
Save As:			
Project			Saves project under another name.
Clean Project			Saves project under another name but does not copy. output and preprocessed files.

Table 24 Project menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Save Selected Experiments As:			
Project			Creates a new project based on the selected experiments. The new project contains the simulation results of the selected nodes.
Clean Project			Creates a new project based on the selected experiments. The new project does not contain the simulation results of the selected nodes.
Search			Opens the Search for Files and Projects dialog box (see Searching for Files and Projects on page 34).
Operations:			
Reload		Ctrl+D	Reloads project.
Stop Loading		Esc key	Stops project loading or reloading.
Preprocess		Ctrl+P	Preprocesses current project.
Preprocess Tcl Blocks		Ctrl+B	Preprocesses Tcl command blocks.
Run		Ctrl+R	Runs current project.
Abort		Ctrl+T	Terminates running project.
Clean Up		Ctrl+L	Shows several project cleanup options.
Rename			Renames project in the projects browser.
Reset Status		Ctrl+K	Resets project status to None.
Convert to Hierarchical			Opens the Convert to Hierarchical dialog box (see Converting Project Organization on page 31).

Table 24 Project menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Properties:			Project properties:
Runtime Editing Mode:			
Locked			Specifies the Locked mode for a project.
Editable			Specifies the Editable mode for a project.
Yes			Creates symbolic links to project files inside each node folder.
No			Does not create symbolic links.
Summary		Ctrl+Y	Shows project summary.
Readme			Opens an editor with the project readme file.
Documentation:			Options for viewing project documentation files.
PDF File			Opens `greadme.pdf` in the default PDF viewer.
HTML File			Opens `greadme.html` in the default browser.
Logs:			Options for viewing project-related logs:
Preprocessor			Views preprocessor log.
Project		Ctrl+J	Views project log.
History		Ctrl+H	Views history log.
Conversion			Views conversion log.
Export			Packages the contents of a project to a `.gzp` file according to user settings in the Export to Package dialog box.
Import			Extracts a zipped or tarred file containing a project into a directory and opens it.

Table 24 Project menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Recent Projects			List of recently opened projects.
Exit		Ctrl+Q	Exits Sentaurus Workbench.

Edit Menu

Table 25 Edit menu commands

Command	Toolbar button	Shortcut keys	Description
Undo		Ctrl+Z	Undoes previous operation.
Cut		Ctrl+X	Cuts a selection of experiments.
Copy		Ctrl+C	Copies a selection of experiments.
Paste		Ctrl+V	Pastes a cut or copied selection of experiments.
Paste Special		Ctrl+M	Allows you to specify what to insert.
Delete		Delete key	Deletes currently selected item.
Attach Root			Attaches a new root in the projects browser.
Detach Root			Detaches root selected in the projects browser.
Tool DB: Global Site User Project			Options for viewing tool database files: Global tool database. Site tool database; available when `SWB_SITE_SETTINGS_DIR` is defined. User tool database. Project tool database.

Table 25 Edit menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Run Limits: Global Site User			Options for viewing run limits settings files: Global run limits settings. Site run limits settings; available when `SWB_SITE_SETTINGS_DIR` is defined. User run limits settings.
Preferences		F12 key	Opens SWB Preferences dialog box.

Scheduler Menu

Table 26 Scheduler menu commands

Command	Toolbar button	Shortcut keys	Description
Show Scheduler			Opens Scheduler.
Configure Queues: User Queues Project Queues Edit User Queues Edit Project Queues			Configures tool associations and node expressions: Configures user queues using user interface. Configures project queues using user interface. Edits user queues using standard text editor. Edits project queues using standard text editor.
View Global Queues			Views the global queue settings in read-only mode.
View Site Queues			Views the site queue settings in read-only mode; available when `SWB_SITE_SETTINGS_DIR` is defined.

View Menu

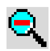
Table 27 View menu commands

Command	Toolbar button	Shortcut keys	Description
Flow Orientation:			Configures simulation flow orientation:
Horizontal			Horizontal orientation (default).
Vertical			Vertical orientation.
Flow View Mode:			Configures simulation flow display:
Full			Displays full simulation flow (default).
Compact			Displays only varying parameters and responses.
Customize Current View			Configures project view by hiding or showing different parts of the simulation flow.
Tree Options:			Submenu for options of viewing tree:
Show Tree		F1 key	When true, shows the simulation tree.
Show Experimental Plan		F2 key	When true, shows the experimental plan.
Show Parameters		F3 key	When true, shows parameter information.
Show Variables		F4 key	When true, shows variable information.
Show Pruned		F8 key	When true, shows pruned nodes.
Show Node Numbers		F9 key	When true, always shows node numbers with parameter values.
Hinting Tool Labels		Ctrl+F9	When true, shows rollover text with the tool labels when the pointer is positioned over the tool icon.
Check Virtual Nodes		Ctrl+0	When true, checks the node status of virtual nodes when running. Distinguishes virtual and real nodes using different colors. Displays virtual nodes in blue.

Table 27 View menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Show Merged Cells			When true, merges node cells with the same parameter value.
Parameter Values		Ctrl+1	Shows parameter values in node cells of the loaded project.
Node Numbers		Ctrl+2	Shows node numbers in node cells.
Host		Ctrl+3	Shows the last host on which the node was run.
Date		Ctrl+4	Shows the time when the last run of the node was completed.
Execution Time		Ctrl+5	Shows the time taken for the last run of the node or an error message.
Variables		Ctrl+6	Shows information about the variables associated with the nodes.
Job Identifier		Ctrl+7	Shows the job identifier assigned to the node when running on a cluster (farm, grid).
Table Options:			Submenu for options of viewing simulation table:
Show Information Titles			Shows the row with titles such as Family Tree and Variable Values.
Show Tool Icons			Shows the row with tool icons.
Show Tool Labels			Shows the rows with tool labels.
Show Comments			Shows the row with tool comments (arbitrary multiline text).
Show Parameter Process Names			Shows the row with the process names of parameters (for process tools only).
Show Parameter and Variable Names			Shows the row with the names of parameters and variables.
Show Experiment Numbers			Shows the column with the numbers of experiments.

Table 27 View menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Change Table Font			Displays a dialog box where you can select the font.
Zoom In		Ctrl+Plus sign	Zooms in to the project view.
Zoom Out		Ctrl+Minus sign	Zooms out of the project view.
Zoom Off		Ctrl+0	Resets zoom to the default.
Freeze Rows/ Columns			Freezes selected rows or columns or both from scrolling
Unfreeze Rows/ Columns			Unfreezes selected rows or columns or both
Restore Default Cell Size		Ctrl+7	Restores default cell size for all project nodes.
Restore Default View Options		Ctrl+8	Restores default view options from preferences.
Refresh		F5 key	Refreshes the tree or browser view.
Properties			Shows the basic properties of the tree.

Scenario Menu

Table 28 Scenario menu commands

Command	Toolbar button	Shortcut keys	Description
Add			Adds a new scenario to the tree.
Delete			Deletes scenarios from the tree.
Properties			Shows basic properties of selected scenario.

Table 28 Scenario menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Next		Alt+ →	Shows next scenario in the tree.
Previous		Alt+ ←	Shows previous scenario in the tree.
Lock			Locks all nodes of current scenario from preprocessing.
Unlock			Unlocks all nodes of current scenario for preprocessing.
Unfold All		Ctrl+Alt+A	Unfolds all folded nodes of the current scenario.
Preprocess			Preprocesses current scenario.

Tool Menu

Table 29 Tool menu commands

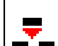
Command	Toolbar button	Shortcut keys	Description
Add		Insert key	Adds a new tool, parameter, experiment, or variable to the tree.
Delete		Delete key	Deletes selected tool, parameter, experiment, or variable from the tree.
Properties			Shows basic properties of selected tool.
Edit Input			Opens submenu for editing input files of selected tool.
SDevice Wizard			Launches the user interface of the Sentaurus Device Wizard for the selected tool. This command is available only for Sentaurus Device tool instances. If this command is not available, then set **Miscellaneous** > **Sentaurus Device Wizard** to **Yes** in the SWB Preferences dialog box.

Table 29 Tool menu commands (Continued)

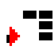
Command	Toolbar button	Shortcut keys	Description
Clean and Synchronize With Parent Project			Synchronizes parameters and variables with a parent project with cleanup of a previous synchronization. This command is available only for the Bridge tool.
Synchronize With Parent Project			Synchronizes parameters and variables missing after the previous synchronization with a parent project. This command is available only for the Bridge tool.
Open Parent Project			Closes current project and opens parent project. If the project has a different organization, Sentaurus Workbench restarts in the correct mode.
Open Parent Project in New SWB Instance			Launches a new Sentaurus Workbench instance with the parent project.
Start Project Database			Starts the project database if it is not yet running. This command is available only for TCAD to SPICE tools.
Stop Project Database			Stops the project database if it is running. This command is available only for TCAD to SPICE tools.
S-Visual API Documentation			Launches Sentaurus Visual API documentation in the default browser. This command is available only for Sentaurus Visual Python tools.
Optimizer API Documentation			Launches Optimizer API documentation in the default browser. This command is available only for the Optimization Framework tool (`genopt`).
TCAD to SPICE API Documentation			Launches TCAD to SPICE API documentation in the default browser. This command is available only for TCAD to SPICE tools.
Mystic API Documentation			Launches Mystic API documentation in the default browser. This command is available only for TCAD to SPICE tools.
Preprocess			Preprocesses all nodes belonging to the selected tools.
Hide			Hides selected tools.

Table 29 Tool menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Show			Shows a hidden tool.
Lock			Locks selected tool with all its nodes.
Unlock			Unlocks selected tool with all its nodes.

Parameter Menu

Table 30 Parameter menu commands

Command	Toolbar button	Shortcut keys	Description
Add Parameter/Values		Insert key	Displays the Add Parameter/Values dialog box where you can add a new parameter with variation to the flow or add values to the selected parameter.
Delete		Delete key	Deletes selected parameter from the flow.
Properties			Shows properties of selected parameter.
Edit Values			Displays dialog box to edit values of the selected parameter.
Remove Value			Removes a value of the selected parameter.
Hide			Hides selected parameters.
Show			Shows a hidden parameter.

Experiments Menu

Table 31 Experiments menu commands

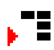
Command	Toolbar button	Shortcut keys	Description
Create Default Experiment			Creates a new experiment for a specified scenario; default values are used for all parameters.
Take Selected Experiment As Default			Changes default values of parameters to the values of the selected experiment.
Add New Experiment		Insert key	Adds a new experiment to the tree (the default values are preset to a selected experiment if there is one).
Add Parameter/Values			Adds parameter values to all experiments (full factorial) or selected experiments.
Manage Membership in Scenarios			Displays the Manage Membership in Scenarios dialog box where you can manage scenarios and associated experiments.
Exclude Experiments		Delete key	Excludes selected experiments from the current scenario.
Delete Experiments			Deletes selected experiments from the flow.
DoE:			Design-of-experiments options (see Chapter 5 on page 145):
DoE Wizard			Opens the DoE Wizard.
Taguchi Wizard			Opens the Taguchi Wizard.
Export:			Submenu for exporting current project view:
Text File			Exports view into a file compatible with the Microsoft Excel application.
Run Spreadsheet Application			Exports view into a file and opens it with a spreadsheet application configurable in preferences.
Import From File			Imports experiments from a text file.

Table 31 Experiments menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Sort Experiments			Sorts experiments according to a defined parameter.
Properties			Shows parameter values of selected experiment.

Nodes Menu

Table 32 Nodes menu commands

Command	Toolbar button	Shortcut key	Description
Select:			Nodes selection submenu:
All		Ctrl+A	Selects all nodes.
By Expression			Selects nodes according to an expression.
By Status			Submenu for selecting nodes by their status; all submitted and all not yet running nodes.
Inverse Of			Submenu for selecting nodes by the inverse of their status or their current selection.
Deselect All		Esc key	Deselects all selected cells.
Extend Selection To:			Submenu for extending selected nodes to:
Root			Root.
Leaves			Leaves.
Experiments			Experiments.
Prerequisite Nodes			Prerequisite nodes.
Edit Value		F6 key	Switches selected node cell to edit mode to allow modifying the value; only single-node selection is supported.

Table 32 Nodes menu commands (Continued)

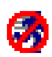
Command	Toolbar button	Shortcut key	Description
Edit Properties			Sets properties of a single selected node or allows you to modify the status of multiple selected nodes.
Modify Multiple Parameter Values			Sets parameter values on multiple nodes at the same time.
Set Variable Value			Sets a variable value for the selected node; creates a variable if it does not exists.
Renumber All Nodes		Ctrl+Alt+R	Renumbers all nodes in the project.
Preprocess			Preprocesses selected nodes.
Run		Ctrl+R	Runs selected nodes.
Quick Run			Runs selected nodes in a selected queue immediately.
Abort		Ctrl+T	Terminates running of selected nodes.
Visualize			Visualizes output files for selected nodes (see Viewing the Output Files of Nodes on page 67). See Table 33 on page 320 for details of the **Nodes** > **Visualize** submenu commands
Quick Visualize			Quickly visualizes output files for selected nodes with the default visualizer.
Node Explorer		F7 key	Displays the Node Explorer.
View Output		Ctrl+W	Views standard output and error files of selected nodes.
Clean Up Node Output		Delete key	Shows cleanup options for selected node.
Configuration: Tool DB Info			Nodes configuration submenu: Shows input and output file names of selected nodes according to the tool database.

Table 32 Nodes menu commands (Continued)

Command	Toolbar button	Shortcut key	Description
Lock			Locks selected nodes from preprocessing and execution.
Unlock			Unlocks selected nodes from preprocessing and execution.
Prune		Ctrl+E	Prunes selected nodes.
Unprune		Ctrl+U	Unprunes selected nodes.
Fold			Folds selected nodes; nodes that do not have child nodes are not folded.
Unfold			Unfolds selected nodes; if a node is not folded, this operation unfolds the nearest preceding folded node (if any).

Table 33 Visualize submenu commands

Command	Toolbar button	Shortcut keys	Description
Sentaurus Visual			Visualizes output files for selected nodes in Sentaurus Visual (see Viewing the Output Files of Nodes on page 67): • **Select File**: Displays a dialog box where you select files for visualization. • **All Files**: Visualizes all supported files.
Sentaurus Visual (Select by Type):			Visualizes specific output files for selected nodes in Sentaurus Visual (see Viewing the Output Files of Nodes on page 67):
All XY-Plot Files			Files with the extensions `.plt` and `.plx`.
All Boundary Files			Files with the extension `*_bnd.tdr`.
All Mesh Files			Files with the extension `*_msh.tdr`.
All TDR Files			Files with the extension `.tdr`.

Table 33 Visualize submenu commands (Continued)

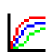
Command	**Toolbar button**	**Shortcut keys**	**Description**
Inspect			Visualizes output files for selected nodes in Inspect: • **Select File**: Displays a dialog box where you select files for visualization. • **All Files**: Visualizes all supported files.
Text			Visualizes text files for selected nodes.
SDE			Visualizes `.sat` files for selected nodes in Sentaurus Structure Editor.
Optimization Plots			Visualizes RSM plots generated by Optimization Framework.
Sentaurus Visual - Sentaurus Process Link			Launches Sentaurus Visual with the Sentaurus Process interface with the process flow of the selected Sentaurus Process node loaded.
XML LogFile			Visualizes `.xml` files for selected nodes in the TCAD Logfile Browser.
HTML LogFile			Visualizes `.html` files for selected nodes in the default browser.
Images			Visualizes `.gif`, `.png`, `.jpg`, and `.jpeg` files for selected nodes in the default image viewer.
Animated Images			Visualizes animated `.gif` files for selected nodes in the default image viewer for animated images (animate).
Run Selected Visualizer Nodes Together			Preprocesses selected visualizer nodes (Sentaurus Visual or Inspect), merges their input command files into one file, and runs the merged file in the visualizer locally (see Viewing Visualizer Nodes Simultaneously on page 72).
Compare Command Files of Selected Nodes			Compares the content of command files of selected nodes using a comparison application (see Comparing Command Files of Nodes on page 74).
PCM Studio: Export Current Scenario			See Table 37 on page 324.

Table 33 Visualize submenu commands (Continued)

Command	**Toolbar button**	**Shortcut keys**	**Description**
PCM Studio: Configure Export			See Table 37 on page 324.

Variables Menu

Table 34 Variables menu commands

Command	**Toolbar button**	**Shortcut keys**	**Description**
Add		Insert key	Adds a new variable to the project.
Delete		Delete key	Deletes selected variables from the project.
Properties			Shows default formula of selected variable.
Format			Displays a dialog box with formatting options for selected variable.
Update Variables			Updates variables view if variables file (`gvars.dat`) has changed on disk.
Hide			Hides selected variables.
Show			Shows a hidden variable.

Optimization Menu

Note:

This menu is shown only in Sentaurus Workbench Advanced mode.

Table 35 Optimization menu commands

Command	Toolbar button	Shortcut keys	Description
Edit Python Input			Opens an editor with the optimization strategy.
Run			Runs the optimization strategy.
Restart From			Restarts the optimization project from a stored scenario.
Reset Project			Resets the optimization project to the initial scenario.
View Log			Opens an editor with the optimization input file.
Optimizer API Documentation			Opens front page to Optimizer API documentation (HTML) in the default browser.
RSM Visualization			Displays the RSM Visualization dialog box.

Calibration Menu

Note:

This menu is shown only in Sentaurus Workbench Advanced mode.

Table 36 Calibration menu commands

Command	Toolbar button	Shortcut keys	Description
Project Wizard			See the *Calibration Kit User Guide*.
Scenario Wizard			See the *Calibration Kit User Guide*.
Process Wizard			See the *Calibration Kit User Guide*.
Parameter Wizard			See the *Calibration Kit User Guide*.
Optimization Wizard			See the *Calibration Kit User Guide*.

PCM Studio Menu

Note:

This menu is shown only when a license for Sentaurus PCM Studio is available. See the *Sentaurus™ PCM Studio User Guide* for more information.

Table 37 PCM Studio menu commands

Command	**Toolbar button**	**Shortcut keys**	**Description**
Export Current Scenario			Exports the current scenario to a temporary CSV file, then starts Sentaurus PCM Studio with this file.
Save Current Scenario to CSV Files			Exports the current scenario to a CSV file <scenario_name>.csv and stores it in the project directory.
Configure Export			Displays dialog box to configure exporting to a CSV file.

Extensions Menu

Table 38 Extensions menu commands

Command	**Toolbar button**	**Shortcut keys**	**Description**
File Browser		F11 key	Browses directory files of project.
Command Prompt Here			Opens a command prompt in the selected project. For a hierarchical project, opens a command prompt in the folder of the selected node.
New Tcl Script			Creates and opens a new file for a Tcl script.
Run Tcl Script			Runs a Tcl script.
Edit Tcl Script			Edits a file containing a Tcl script.
New Python Script			Creates and opens a new file for a Python script.

Table 38 Extensions menu commands (Continued)

Command	Toolbar button	Shortcut keys	Description
Run Python Script			Runs a Python script.
Edit Python Script			Edits a file containing a Python script.
Run Sentaurus Structure Editor			Launches Sentaurus Structure Editor.
Run Sentaurus Process			Launches Sentaurus Process.
Run Sentaurus Visual			Launches Sentaurus Visual.
Run Inspect			Launches Inspect.
Run Log Browser			Launches TCAD Logfile Browser to view `.xml` and `.html` log files of the selected node.
Run Sentaurus Visual - Sentaurus Process Link			Launches Sentaurus Visual with Sentaurus Process interface where you can load the process flow of the selected Sentaurus Process node and interactively run it.
Run Layout Viewer			Launches the layout viewer as specified in the preferences: **Utilities** > **Layout Viewer**.
Reconstruct Extracted Results			Parses all the node output files `n*_*.out`, finds the extracted results, and updates the Variables field in the project.
Save Extracted Results Summary			Parses all the node output files `n*_*.out`, finds the extracted results, and saves them in the given `.csv` file.

Help Menu

Table 39 Help menu commands

Command	Toolbar button	Shortcut keys	Description
About		Ctrl+B	Provides version information.
Manuals			Opens front page to technical documentation (PDF) in default PDF viewer.
Python API Documentation: SWB S-Visual TCAD to SPICE Mystic Optimizer			Opens front page to Python API documentation (HTML) for a given tool in default browser: Sentaurus Workbench Python API Sentaurus Visual Python API TCAD to SPICE Python API Mystic Python API Optimizer Python API
Training			Opens TCAD Sentaurus Tutorial (HTML training material) in default browser.

Toolbar Buttons of Project Editor

Table 40 Project Editor toolbar buttons

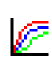
Toolbar button	Description	Toolbar button	Description
	Creates a new project.		Quickly visualizes output files for selected nodes with the default visualizer.
	Converts traditional project to hierarchical project.		Runs selected visualization nodes together.
	Opens a project.		Displays information (properties) about the currently selected item.

Table 40 Project Editor toolbar buttons (Continued)

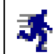
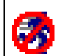
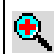
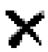
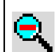
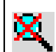
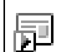
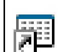
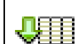
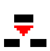
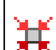
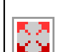
Toolbar button	Description	Toolbar button	Description
	Saves a project under an existing name.		Runs current project or selected nodes.
	Reloads a project.		Terminates running project or selected nodes.
	Stops the loading of a project.		Zooms in to the project view.
	Closes the currently opened project.		Zooms out of the project view.
	Cuts a selection.		Resets zoom to the default.
	Copies a selection.		Changes the current project view to the next view.
	Pastes a cut or copied selection.		Opens a spreadsheet application with the current view.
	Undoes previous operation.		Displays a simulation project vertically.
	Adds a new tool, parameter, experiment, or variable to the tree.		Displays a simulation project horizontally.
	Deletes selected tool, parameter, experiment, or variable from the tree.		Displays a simulation project in compact mode.
	Adds a new experiment to the tree (the default values are preset to a selected experiment if there is one).		Displays a simulation project in full mode.
	Adds parameter values to either all (full factorial) or selected experiments.		Opens a command prompt in a project directory as a separate shell. For a hierarchical project, opens a command prompt in the folder of the selected node.
	Opens submenu for editing input files of selected tool.		Opens manuals in PDF format in default PDF viewer.

Table 40 Project Editor toolbar buttons (Continued)

Toolbar button	Description	Toolbar button	Description
	Visualizes output files for selected nodes.		Opens TCAD Sentaurus Tutorial (HTML training material) in default browser.

Keyboard Navigation Keys

Table 41 Keyboard navigation keys

Key	Description
Tab	Moves between the projects browser and project viewer panel.

C

Sentaurus Workbench Files

This appendix lists the different files relevant to Sentaurus Workbench.

Project Files

The following project files are specific to Sentaurus Workbench:

`gcomments.dat`	Contains comments for project tool steps.
`genopt.py`	Input file of Optimization Framework.
`gexec.cmd`	Jobs execution graph.
`glog.txt`	Contains the project log data.
`gopt.log`	Log file of Optimization Framework.
`gqueues.dat`	Queues configuration file.
`greadme.pdf`	The documentation file of a project.
`greadme.txt`	The readme file of a project.
`gsummary.txt`	Summary of a project.
`gtooldb.tcl`	The tool database of a project.
`gtree.dat`	Contains the simulation tree.
`gvars.dat`	Contains all variables of a project.
`preprocessor.log`	Contains preprocessor log data.
`runlimits.txt`	Contains project run limits.

Hidden Files

The following hidden files are not necessarily visible in the directory listings of a project:

`.database`	Database of project view settings.
`.history`	Events history of a project (including preprocessing, runs, and cleanup).
`.organization`	Project organization, either traditional or hierarchical.
`.project`	Empty file indicating that the current directory is a project directory.
`.status`	Project status.

User Configuration Files

All these files are in the `env(STDB)` directory:

`gqueues_<user>.dat`	Queues configuration file.
`tooldb_<user>`	The tool database of the user.
`runlimits_<user>.xml`	Run limits settings of the user.
`gpref2_<user>.<release>.xml`	Preference file. There is one preference file for each release of TCAD Sentaurus, including feature and service pack releases. For example: • Sentaurus Workbench, Version S-2021.06, would use the `gpref2_jsmith.S-2021.06.xml` file. • Sentaurus Workbench, Version S-2021.06-SP1, would use the `gpref2_jsmith.S-2021.06-SP1.xml` file.

Global Configuration Files

These files are in the following directories:

- `$STROOT/queues; $STROOT/tcad/$STRELEASE/lib/glib2`:

  `gqueues.dat` Queues configuration file.

- `$STROOT/tcad/$STRELEASE/lib/glib2`:

`tooldb.tcl`	Global tool database.
`runlimits.xml`	Global run limits settings.
`gpref2.$STRELEASE.xml`	Global preferences.

Site Configuration Files

These files are in the following directories:

- `SWB_SITE_SETTINGS_DIR`:

`gqueues.dat`	Queues configuration file.
`tooldb.tcl`	Site tool database.
`runlimits.xml`	Site run limits settings.
`gpref2.$STRELEASE.xml`	Site preferences.

Typical Input and Output Files

These are some typical file patterns seen in projects. The actual file names depend on the definitions of the tool database:

`<tool_label>_<tool_acronym>.cmd`	Input file template: the command file of a tool.
`<tool_label>_<tool_acronym>.*`	The other input files of a tool that are configurable in the tool database.
`pp<node_number>_<tool_acronym>.*`	Node input files created by preprocessing.
`n<node_number>_<tool_acronym>.*`	Node output files.

D

Known Issues on VNC Clients

This appendix discusses known issues related to running Sentaurus Workbench on VNC clients.

Double-Clicking Operation Does Not Work

On some VNC clients, double-clicking a node, parameter, tool, or variable does not work as expected. For example, if the action of double-clicking a tool is bound to editing the tool input file in the preferences, the expected text editor does not open after double-clicking.

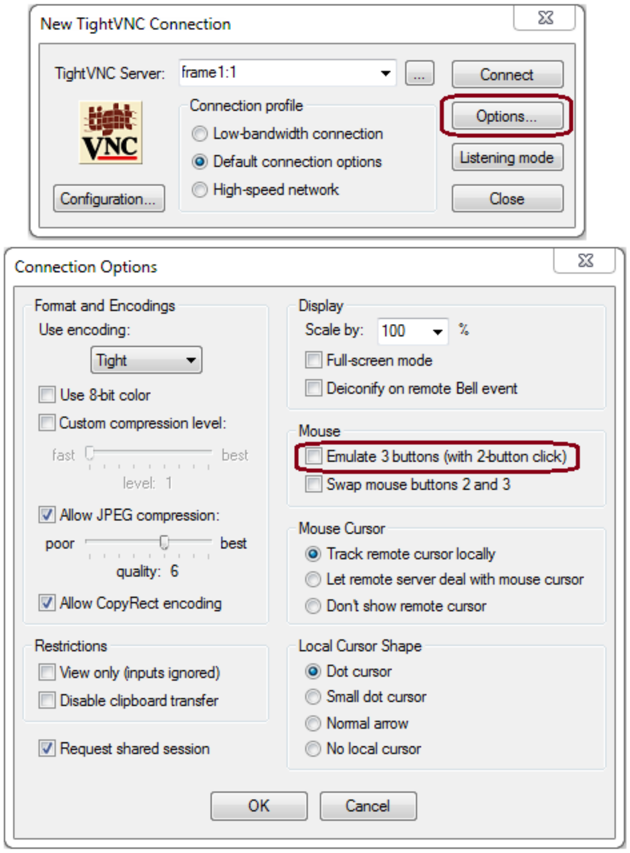
In particular, this problem appears on the TightVNC client where the double-clicking operation is prevented by the default mouse settings on the TightVNC client.

Workaround

In the Connection Options dialog box (see Figure 83 on page 333), the **Emulate 3 buttons (with 2-button click)** option is selected by default, which results in incorrect double-clicking action in Sentaurus Workbench. This option must be cleared.

This problem has not been observed on other VNC clients, in particular, RealVNC.

Figure 83 TightVNC connection options



E

Troubleshooting Network Issues

This appendix describes how to diagnose network issues affecting the communication between Sentaurus Workbench components.

Configuring Sentaurus Workbench Behind a Firewall

Sentaurus Workbench communication architecture uses TCP ports for sending messages between processes. There are no predefined port numbers, so Sentaurus Workbench takes TCP ports that are not currently used by other processes.

Some Synopsys customers work under strict firewall restrictions. It might be that all of the TCP ports, except a few, are closed and cannot be used for message communications. If this is the case, then you must contact your corporate IT department to obtain a list of allowed TCP ports. Then, you need to communicate this information to Sentaurus Workbench using a comma list in the environment variable `SWB_PORTS_RANGE`. Each part of the comma list must contact either a specific port or a range of ports. For example:

```
setenv SWB_PORTS_RANGE 40000-40010

setenv SWB_PORTS_RANGE 40000-40010,40020,40050-40060

setenv SWB_PORTS_RANGE 30000,4000,40100,40200,40300
```

When Sentaurus Workbench needs to open a TCP port, it checks the given range of ports and takes the first one that is free.

You can evaluate the number of TCP ports required for Sentaurus Workbench on a host using the following formula:

$N_{ports} = 1 + N_{swb} + N_{projects}$

where N_{swb} is the number of Sentaurus Workbench sessions you run on this host, and $N_{projects}$ is the number of simultaneously running projects you launch from within these Sentaurus Workbench sessions.

Sentaurus Workbench Diagnostics Tool

Synopsys' customers work in differently configured distributed environments. It might be that networking issues or incorrect environment settings affect Sentaurus Workbench operations.

Investigations of various issues that customers have experienced working with Sentaurus Workbench have shown that most of these issues seem to be due to specific operating system configurations or network issues. Examples include, but are not limited to:

- TCP ports of the host are closed.
- Host A does not see host B with the `ping` command, and so on.
- A significant network latency.

The Sentaurus Workbench Diagnostics tool checks whether the Sentaurus Workbench infrastructure can run properly on two given hosts, so that detected issues can be addressed promptly from the customer side.

Troubleshooting the Sentaurus Workbench Network

To test and debug possible issues with the communication architecture of Sentaurus Workbench, the Sentaurus Workbench Diagnostics tool must be executed.

This batch tool detects and addresses specific issues in the working environment that can affect the behavior of Sentaurus Workbench. In particular, this tool checks that the:

1. Sentaurus Workbench infrastructure can communicate among two or more hosts.
2. Sentaurus Workbench environment variables `STROOT`, `STRELEASE`, `STROOT_LIB`, and `STDB` are set correctly.

For item 1, given a list of machines, the tool iterates through all combinations, starting the `swblm` daemon process on one machine and having all other machines, iteratively, trying to *ping* it and, if successful, trying to connect to the daemon.

In more detail, the steps for item 1 are:

1. Select a machine (the server) to host the `swblm` daemon and connect to it.
2. Terminate any previous instance of `swblm` on the host machine.

   You can also do this by running the following command on any host: `swblm -cleanup`
3. Set up the advanced logging mode for `swblm` as follows:

   ```
   setenv SWBLM_ADVANCED_LOG 1
   ```

4. Start a new `swblm` daemon.
5. Check for the `~/.swb/swblm/<release>/swblm.conf` file.
6. Iterate through the rest of the machines and:
   a. Connect to the selected machine (the client).
   b. Ping from the client machine to the server machine.
7. Request a connection to the `swblm` daemon on the server machine.
8. Return to Step 1.

For item 2, the tool checks the following environment variables:

- `STROOT`: It issues a warning if the variable is not set.
- `STRELEASE`: It issues a warning if the variable is not set.
- `STROOT_LIB`:
  - It does not issue a warning if the variable is not set. Typically, users do not set this variable.
  - If you set this variable, the tool issues a warning if the variable is *not* set to the default value of `$STROOT/tcad/$STRELEASE/lib`, since there might be a library mismatch between different TCAD Sentaurus releases.
- `STDB`: It issues a warning if the variable is not set.

Limitations and Assumptions

The Sentaurus Workbench Diagnostics tool does not check the validity of host names. It assumes these follow the proper rules of the system.

The entire Sentaurus Workbench infrastructure must be switched off during such tests since the `swblm` daemon must be restarted. You must stop all your running projects and exit all Sentaurus Workbench instances of the current release.

A silent remote connection is required (no password is needed or other interactive authentication) in the case of remote host checking. See Appendix F on page 339 for instructions on how to configure SSH without a password.

It is assumed that all checked hosts share the file system, which is also the assumption for Sentaurus Workbench operations.

Usage

To start the test, execute the following command in the shell:

```
% swbdiag [options] [HOSTS]
```

The options are:

`-h[elp]`

Displays details about how to use the Sentaurus Workbench Diagnostics tool.

`-v[ersion]`

Displays the current version of the tool.

`-f[ile]`

Allows you to specify a file containing a list of hosts to test. If this option is defined, any list of hosts specified outside the file is ignored.

Instead of a file, you can directly provide a list of hosts as arguments. The following example illustrates the differences:

- Hosts specified through a file:

  ```
  % swbdiag -f hosts.txt
  ```

- Hosts specified as direct arguments:

  ```
  % swbdiag hostA rsh:hostB ssh:hostC
  ```

If the list of hosts is provided in a file, it must be a plain text file containing the name of the hosts separated by space, tabs, or newlines.

The Sentaurus Workbench Diagnostics tool requires and accepts the `rsh:` and `ssh:` prefixes to specify the connection type to the remote host. The tool assumes that hosts without a prefix correspond to the local host.

Report and Log File

The Sentaurus Workbench Diagnostics tool prints to the screen the current step of the test. If the tool successfully terminates any previous instance of the `swblm` daemon (item 1, Step 2), at the end of the test, a report is generated and saved to:

```
~/.swb/swblm/<RELEASE>/swblm.test
```

This file contains exactly the same information that has been printed to the screen.

The report is delimited as follows:

- Each test to check the Sentaurus Workbench infrastructure (item 1) is delimited by two lines composed of the hash (#) character.
- Each connection test between the local host and the remote host is delimited by one line composed of the equal sign (=).
- At the end of the test, the Sentaurus Workbench environment variables are checked. The results are printed immediately after the last pair of lines composed of the # character:
  - If everything is acceptable with a variable, you should see only the following message:

    `<Timestamp> - INFO: Checking variable <Variable under test>`

  - If the case is not set, a warning is issued.
  - The only exception is `STROOT_LIB` since it is not a problem if this environment variable is not set (only an information message is printed). However, a warning is issued if the variable is set but does not match the following path:

    ```
    $STROOT/tcad/$STRELEASE/lib
    ```

If an error is detected during the testing of the Sentaurus Workbench infrastructure, a summary table is printed at the end of the log file. The report groups the error messages into the following categories:

- Initialization: `SWBLM could not be initialized`
- Release mismatch: `SWBLM release version mismatch or not found`
- Configuration file: `SWBLM configuration file not found or corrupted`
- Connection: `These hosts could not connect to server/SWBLM or host does not exist`

Here is an example of the report:

```
Server       Issue

us01acme01   SWBLM could not be initialized
cl01narf03   SWBLM release version mismatch or not found
cn04kung12   SWBLM configuration file not found or corrupted
us03acmerh05 These hosts could not connect to server/SWBLM or host does
             not exist: * us03acmerh01,* us03acmerh02,* us03acmerh03
```

The Server column indicates the machine that it is hosting the `swblm` daemon during that stage of the test.

F

Using SSH

This appendix provides information about using SSH without a password and troubleshooting tips.

Configuring SSH Without a Password

To configure SSH without a password:

1. Generate the SSH key.

   Note:
   Leave the passphrase empty when prompted as shown in the following example.

   ```
   [pjones@myunix.com ~]$ ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/pjones/.ssh/id_rsa):
   [Press the Enter key]
   Created directory '/home/pjones/.ssh'.
   Enter passphrase (empty for no passphrase): [Press the Enter key]
   Enter same passphrase again: [Press the Enter key]
   Your identification has been saved in /home/pjones/.ssh/id_rsa.
   Your public key has been saved in /home/pjones/.ssh/id_rsa.pub.
   The key fingerprint is:
   5f:ad:40:00:8a:d1:9b:99:b3:b0:f8:08:99:c3:ed:d3 pjones@myunix.com
   ```

2. Copy the new public SSH key into the `authorized_keys` file in your home directory. For example:

   ```
   [pjones@myunix ~]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

3. Update or create the SSH `config` file in your home directory. For example:

   ```
   [pjones@myunix ~]$ vi ~/.ssh/config
      Host *
         StrictHostKeyChecking no
         UserKnownHostsFile /dev/null
         IdentityFile ~/.ssh/id_rsa
   ```

4. Check that you can use SSH without a password.

Troubleshooting Tips

SSH is very sensitive about file permissions. For example:

```
a. ~/.ssh/authorized_keys file needs to be 640
-rw-r----- 1 pjones synopsys    401 Jan 18  2022 authorized_keys

b. ~/.ssh directory needs to be 700
rwx------ 4 pjones synopsys 4096 Jan 18     2022 /home/pjones/.ssh

c. ~/.ssh/config needs to be 644
-rw-r--r-- 1 pjones synopsys     91 Jan 18  2022 config

d. ~/.ssh/id_rsa (private key) needs to be 600
-rw------- 1 pjones synopsys   1675 Jan 18  2022 id_rsa

e. The owner group must not have write permissions on your home
directory. Chmod your home directory to 755 / drwxr-xr-x
```

If you experience errors with the message *Agent admitted failure to sign using the key*, then try setting `SSH_AUTH_SOCK=0` in your environment and try SSH again.

If this resolves the issue, then set this in your `.bashrc` or `.cshrc` file.

G

Configuring Parallel Environments in SGE Scheduler

This appendix describes how to configure parallel environments in the SGE scheduler for running SMP and MPI jobs.

Parallel Environments

Submitting parallel jobs to the SGE scheduler requires configuring parallel environments on the SGE side. Sentaurus Workbench uses SGE parallel environments for the proper allocation of cores when submitting parallel jobs. For SMP jobs, the cores for multithreaded processes must be allocated on the same host. Processes of an MPI job should be allocated according to the specified processes-per-host distributions, while all threads of every single MPI worker process must run on the same host.

Unlike LSF or other schedulers, parallel environments in the SGE scheduler are not standardized and can be configured in a user-specific way. For example, the Univa Grid Engine farm in Synopsys configures the following parallel environments:

```
mt, dp, dp2, dp3, dp4, dp5, dp6, dp7, dp8, dp10, dp12, dp16, dp20,
dp24, dp32, dp40
```

Use the following SGE command to return a list of available SGE parallel environments:

```
$ qconf -spl
```

Table 42 summarizes the purposes of different parallel environments.

Table 42   Parallel environments

Name of parallel environment	Purpose	Submission example
`mt`	Allocates cores for multithreaded (SMP) jobs.	`$ qsub -pe mt 4 ...` Requests four cores on the same host.
`dp`	Allocates cores for distributed (MPI) jobs with serial MPI worker processes with the default processes-per-host distribution.	`$ qsub -pe dp 4 ...` Requests four cores on arbitrary hosts, and the SGE scheduler determines how to distribute processes. By default, the SGE scheduler tries to allocate all cores on individual hosts. However, the exact behavior depends on the SGE scheduler configuration.
`dp<NN>`	Allocates cores for distributed (MPI) jobs with multithreaded MPI worker processes or specific processes-per-host distribution. Here, `<NN>` is a number starting from 2.	`$ qsub -pe dp4 64 ...` Requests 64 cores on arbitrary hosts, but every four cores must be allocated on the same host. By default, the SGE scheduler tries to allocate all four-core blocks on individual hosts, if possible.

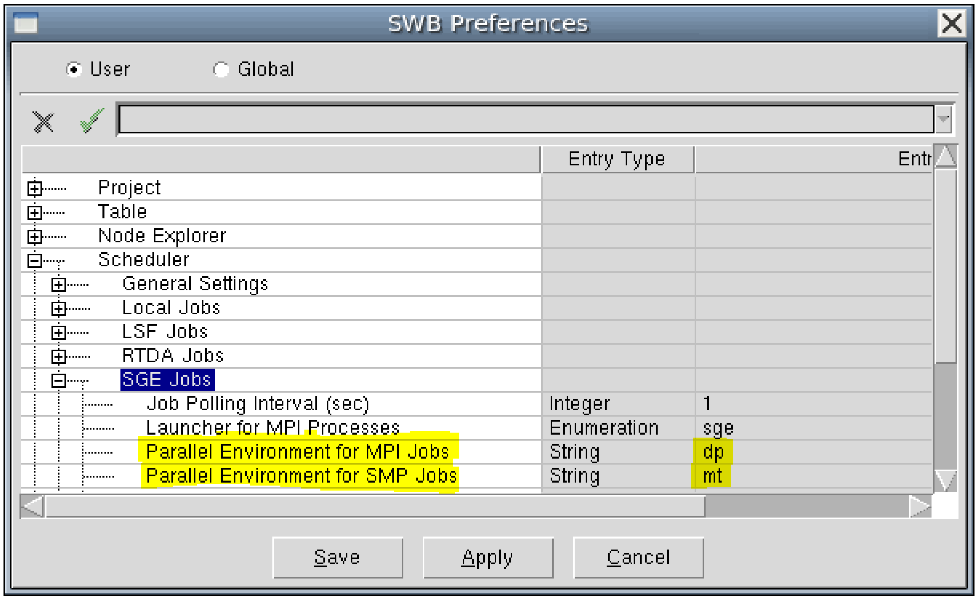
Sentaurus Workbench automates the resource allocation on the SGE grid when submitting an MPI job, based on the number of MPI processes, the number of threads per MPI process, and the MPI processes distribution on hosts.

The names `mt` and `dp` are suggested but you can use another naming scheme for SGE parallel environments. The only requirement from the Sentaurus Workbench side is that the parallel environments for distributed processing must all be named with the same prefix, for example, `mpi_`, `mpi_2`, `mpi_4`.

Using the SWB Preferences dialog box, you can specify the default SGE parallel environments as `mt` and `dp` (see Figure 84 on page 343).

If your SGE farm configures parallel environments with different names, then you must replace the default names with the actual ones.

Figure 84 SWB Preferences dialog box showing default names for SGE parallel environments



Basic SGE Parallel Environments

The following configurations are recommended for basic SGE parallel environments.

mt

```
pe_name                 mt
slots                   9900000
user_lists              NONE
xuser_lists             NONE
start_proc_args         NONE
stop_proc_args          NONE
per_pe_task_prolog      NONE
per_pe_task_epilog      NONE
allocation_rule         $pe_slots
control_slaves          FALSE
job_is_first_task       TRUE
urgency_slots           min
accounting_summary      FALSE
daemon_forks_slaves     FALSE
master_forks_slaves     TRUE
```

dp

```
pe_name                 dp
slots                   999999
user_lists              NONE
xuser_lists             NONE
start_proc_args         /bin/true
stop_proc_args          /bin/true
per_pe_task_prolog      NONE
per_pe_task_epilog      NONE
allocation_rule         $round_robin
control_slaves          TRUE
job_is_first_task       FALSE
urgency_slots           min
accounting_summary      FALSE
daemon_forks_slaves     FALSE
master_forks_slaves     FALSE
```

dp2

```
pe_name                 dp2
slots                   999
user_lists              NONE
xuser_lists             NONE
start_proc_args         /bin/true
stop_proc_args          /bin/true
per_pe_task_prolog      NONE
per_pe_task_epilog      NONE
allocation_rule         2
control_slaves          TRUE
job_is_first_task       TRUE
urgency_slots           min
accounting_summary      FALSE
daemon_forks_slaves     FALSE
master_forks_slaves     FALSE
```

dp3

```
pe_name                 dp3
slots                   999999
user_lists              NONE
xuser_lists             NONE
start_proc_args         /bin/true
stop_proc_args          /bin/true
per_pe_task_prolog      NONE
per_pe_task_epilog      NONE
allocation_rule         3
control_slaves          TRUE
job_is_first_task       TRUE
urgency_slots           min
accounting_summary      FALSE
daemon_forks_slaves     FALSE
master_forks_slaves     FALSE
```

Additional `dp`-based parallel environments are similar to `dp2` and `dp3`, with corresponding changes in the `pe_name` and `allocation_rule` fields.

Adding a Parallel Environment to an SGE Farm

Note:

These operations require administrator privileges on the SGE grid.

To add a parallel environment to an SGE farm:

1. Store the corresponding parallel environment settings in a file.
2. Add the parallel environment to the farm configuration by using the following command:
   ```
   $ qconf -Ap <filename>
   ```
3. Add the created parallel environments to the queue definition of the farm.

   You must add the parallel environments to the list (`pe_list`) of each queue in which you intend to use them. Use the command to add the list to your queue definition:
   ```
   $ qconf -mattr queue pe_list <pe_name> <queue_name>
   ```
   Alternatively, you can use the following command to add the list using a text editor:
   ```
   $ qconf -mq <queue_name>
   ```
   For example:
   ```
   $ qconf -mattr queue pe_list dp2 all.q
   ```

After you complete these steps, you can use the parallel environment in your SGE farm.

You must apply these steps for every parallel environment you want to add to your SGE farm.