Specador Documentation Generator User Guide

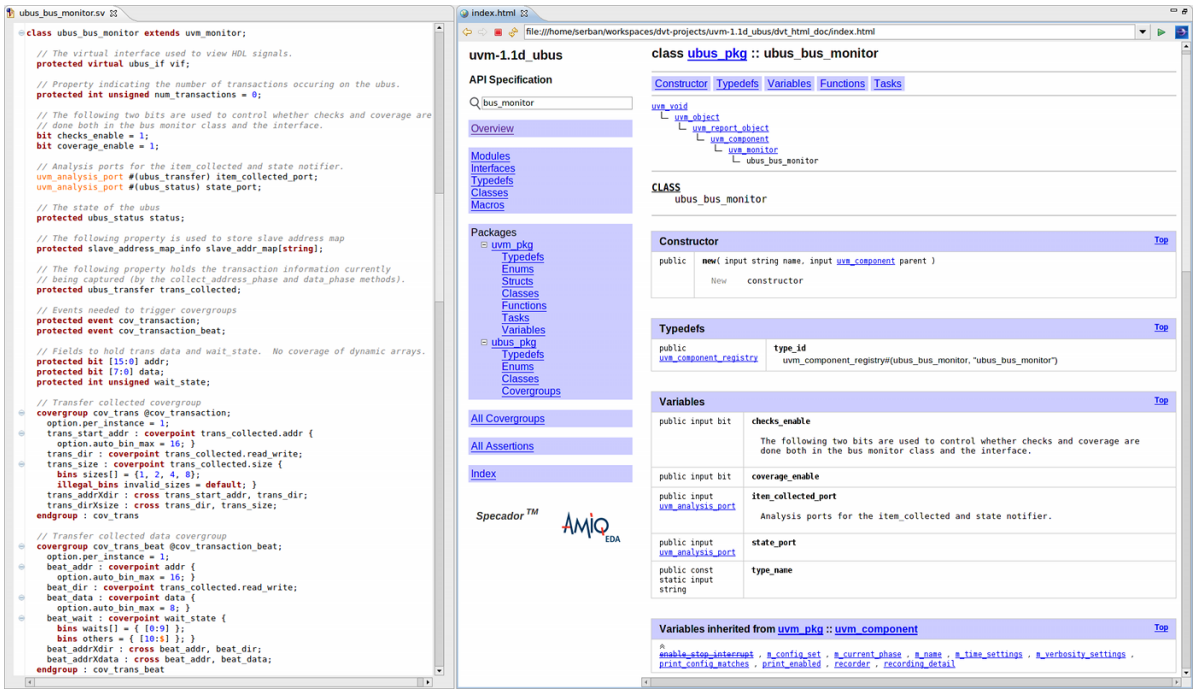
Rev. 17.1.12
26 May 2017

Technical Support: support@amiq.com

Table of Contents

Chapter 1. Overview

Specador automatically generates accurate HTML documentation from e, SystemVerilog, Verilog and VHDL source code by using dedicated language parsers. It enables the user to easily generate and maintain well organized and consistent documentation based on the comments in the source code.



Chapter 2. How to Run

Specador can be invoked in batch mode by running:

$DVT_HOME/bin/specador.sh ...

Examples:

```
 $> specador.sh -lang vlog -cmd /path/to/simulation.f -title "MY CHIP"

 $> specador.sh -lang vhdl -cmd /path/to/file_list.f -preferences /path/to/dvt_export_html.xml

 $> specador.sh -lang e -ignore_compile_errors -cmd irun.args -title "USB 3.0 uVC"
```

Arguments

-lang e|vlog|vhdl
  The source code language: e Language, SystemVerilog or VHDL.
-cmd <file>
  The compilation command file.
[-ignore_compile_errors]
   Ignore compile errors and continue.
[-preferences <XML file>]
  Use preferences specified in the XML file.
[-menu <XML file>]
  Use menu specified in the XML file. It has precedence over the html menu specified by preferences.
[-title <title>]
  Use specified title. It has precedence over the title specified by preferences.

NOTE: When generating HTML documentation in GUI mode, a *dvt_export_html.xml* settings file is saved in the project's *.dvt* directory.

NOTE: In order to generate diagrams, make sure the Graphviz **dot** executable is installed & accessible.

Chapter 3. XML Settings File Syntax

```
<!-- The XML file header, required. -->
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE spechtml PUBLIC "-//DVT//specador-preferences" "specador-preferences.dtd">
<spechtml version="6">

  <!-- GENERAL OPTIONS -->

  <!-- Location where documentation will be generated. Relative paths are solved as relative to the current di
  <location>dvt_html_doc</location>

  <!-- Delete all files in the destination directory before generating documentation. -->
  <clean-location>false</clean-location>

  <!-- When running in GUI mode, choose whether to open or not the generated documentation in browser. -
  <open-in-browser>true</open-in-browser>

  <!-- Title -->
  <title>Documentation Title</title>

  <!-- The content of the overview file will be embedded in the first page.
  Relative paths are solved as relative to the current directory. -->
  <overview-file>README.TXT</overview-file>

  <!-- Beautify comments: start sentences with capital letter, append dot after sentences, bfm -> BFM, dut ->
  <enhance-comments>true</enhance-comments>

  <!-- Generate documentation using the new HTML style -->
  <use-new-html-style>true</use-new-html-style>

  <!-- Syntax for comments formatting: auto, naturaldocs, javadoc or none. -->
  <doc-formatting-type>auto</doc-formatting-type>

  <!-- The user defined navigation menu (HTML or XML format) will be embedded in the main menu.
  Relative paths are solved as relative to the current directory. Default: none. -->
  <user-defined-html>user_defined.html</user-defined-html>

  <!-- Add "Created by <username> ..." watermark. -->
  <created-by-user-watermark>true</created-by-user-watermark>

  <!-- Generate documentation only for public API. -->
  <public-only>true</public-only>

  <!-- API matching the below filters with not be included in the generated documentation. Default: none. --
  <!-- You can specify a comma-separated list of name patterns. Patterns may contain: * = any string, ? = any
  <filter-string>type1, type*, ty?pe</filter-string>

  <!-- You can specify one or more file or directory paths. -->
  <filter-path>/filter/path1</filter-path>
```

```
<filter-path>/filter/path2</filter-path>

<!-- Generate documentation only for files physically located under project root. Relevant for GUI mode or
<just-project-files>false</just-project-files>

<!-- Generate the diagrams only if the number of nodes does not exceed this threshold value. Limitation: w
<diagram-max-nof-nodes>50</diagram-max-nof-nodes>

<!-- Generate design block diagram for each module, entity. -->
<export-design-block-diagram>false</export-design-block-diagram>

<!-- Generate design flow diagram for each module, entity. -->
<export-design-flow-diagram>false</export-design-flow-diagram>

<!-- Generate design schematic diagram for each module, entity. -->
<export-design-schematic-diagram>false</export-design-schematic-diagram>


<export-fsm-diagrams>false</export-fsm-diagrams>

<!-- CROSS-LINK WITH PRE-GENERATED DOCUMENTATION -->

<!-- Link to the elements for which documentation is already generated in the directories specified below, i
<external-doc-path>/path/to/external/doc1</external-doc-path>
<external-doc-path>/path/to/external/doc2</external-doc-path>

<!-- Add entries in the main menu with links to the external documentation. -->
<show-external-doc-refs-in-navbar>false</show-external-doc-refs-in-navbar>

<!-- SYSTEMVERILOG SPECIFIC -->

<!-- Generate modules documentation. -->
<export-vlog-modules>true</export-vlog-modules>

<!-- Generate interfaces documentation. -->
<export-vlog-interfaces>true</export-vlog-interfaces>

<!-- Generate programs documentation. -->
<export-vlog-programs>true</export-vlog-programs>

<!-- Generate macros documentation. -->
<export-vlog-macros>true</export-vlog-macros>

<!-- Generate ifndef guards documentation. Default: false. -->
<export-vlog-ifndef-guards>true</export-vlog-ifndef-guards>

<!-- Generate control defines documentation. Default: true. -->
<export-vlog-control-defines>false</export-vlog-control-defines>

<!-- Generate elements in the global scope (typedefs, classes, functions, tasks etc.). -->
<export-vlog-global-scope>true</export-vlog-global-scope>
```

```
  <!-- Generate packages documentation. -->
  <export-vlog-packages>true</export-vlog-packages>

  <!-- Generate specific package documentation. -->
  <export-vlog-package>uvm_pkg</export-vlog-package>
  <export-vlog-package>ubus_pkg</export-vlog-package>

  <!-- Generate assertions documentation. -->
  <export-vlog-assertions>true</export-vlog-assertions>

  <!-- Generate covergroups documentation. -->
  <export-vlog-covergroups>true</export-vlog-covergroups>

  <!-- Generate interface signals documentation. -->
  <hide-vlog-interface-signals>false</hide-vlog-interface-signals>

  <!-- Generate UML inheritance diagram for each class. -->
  <export-vlog-inheritance-diagram>false</export-vlog-inheritance-diagram>

  <!-- Generate UML direct associations diagram for each class. -->
  <export-vlog-direct-associations-diagram>false</export-vlog-direct-associations-diagram>

  <!-- Generate UML collaboration diagram for each class. -->
  <export-vlog-collaboration-diagram>false</export-vlog-collaboration-diagram>

  <!-- Generate UML inheritance diagram for all classes in each package. -->
  <export-vlog-classes-inheritance-diagram>false</export-vlog-classes-inheritance-diagram>

  <!-- Generate module flow diagram for each module. -->
  <export-vlog-module-diagram>false</export-vlog-module-diagram>

  <!-- Generate module flow diagram including ports for each module. -->
  <export-vlog-module-diagram-with-ports>false</export-vlog-module-diagram-with-ports>

  <!-- VHDL SPECIFIC -->

  <!-- Generate libraries documentation. -->
  <export-vhdl-libraries>true</export-vhdl-libraries>

  <!-- Generate specific library documentation. -->
  <export-vhdl-library>ieee</export-vhdl-library>
  <export-vhdl-library>std</export-vhdl-library>

  <!-- Generate architecture flow diagram for each entity. -->
  <export-vhdl-architecture-flow-diagram>false</export-vhdl-architecture-flow-diagram>

  <!-- Generate architecture flow diagram including ports for each entity. -->
  <export-vhdl-architecture-flow-diagram-with-ports>false</export-vhdl-architecture-flow-diagram-with-por

  <!-- E LANGUAGE SPECIFIC -->

  <!-- Generate macro documentation. -->
```

```
  <export-e-macro>true</export-e-macro>

  <!-- Generate packages documentation. -->
  <export-e-packages>true</export-e-packages>

  <!-- Generate specific package documentation. -->
  <export-e-package>main</export-e-package>
  <export-e-package>vt</export-e-package>

  <!-- Generate UML inheritance diagram for each struct and unit. -->
  <export-e-inheritance-diagram>false</export-e-inheritance-diagram>

  <!-- Generate UML direct associations diagram for each struct and unit. -->
  <export-e-direct-associations-diagram>false</export-e-direct-associations-diagram>

  <!-- Generate UML collaboration diagram for each struct and unit. -->
  <export-e-collaboration-diagram>false</export-e-collaboration-diagram>

  <!-- Generate UML inheritance diagram for all structs and units in each package. -->
  <export-e-package-inheritance-diagram>false</export-e-package-inheritance-diagram>

</spechtml>
```

Chapter 4. XML Menu File Syntax

You can create your own custom navigation menu using a settings directive like:

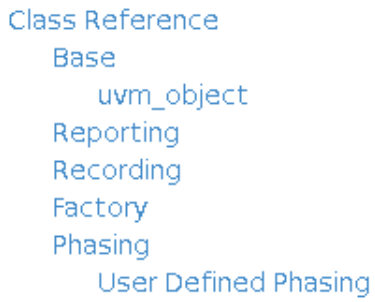
<user-defined-html>my_menu.xml</user-defined-html>

The XML menu syntax is:

```
<!-- The XML file header, required. -->
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<menu version="1"        <!-- Optional. Syntax version.
                         Default latest. -->

 title="Intro"           <!-- Optional. A title for the menu root node.
                         Default basename, if basename.txt is specified for src. -->

 position="top"          <!-- Optional. Menu position.
                         Default bottom. -->

 src="some.txt"          <!-- Optional. A text file to process (recognize NaturalDocs or Javadoc syntax).
                         Default none. -->

 dest="some.html"        <!-- Optional. A file to link the menu entry to.
                         Default basename.html, if basename.txt is specified for src.
                         Otherwise no link. -->

 defaultPage="some.html" <!-- Optional. The default content page when opening index.html.
                         By default is the overview page, if any. -->

 overview="false"        <!-- Optional. Add Overview menu entry, unless false.
                         By default the Overview menu entry points to the Overview page.
                         The Overview page can be generated from the specified overview-file. -->
 >

   <item
    title="1 Menu"     <!-- Optional. A title for the menu node.
                         Default basename, if basename.txt is specified for src. -->

    src="chap1.txt"    <!-- Optional. A text file to process (recognize NaturalDocs or Javadoc syntax).
                         Default none. -->

    dest="ch1.html"    <!-- Optional. A file to link the menu entry to.
                         Default basename.html, if basename.txt is specified for src.
                         Otherwise no link. -->
    >
     <item src="a.txt" title="1.1 Submenu"/>
     <item src="b.txt" title="1.2 Submenu">
       <item src="a.txt" title="1.2.1 Submenu"/>
     </item>
```

```
    </item>
</menu>
```

For example you get a menu like:



from:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<menu version="1" title="Class Reference" src="docs/html/src/overviews/intro.txt" position="top" defaultPa
  <item src="docs/html/src/overviews/base.txt" title="Base">
    <item dest="uvm_pkg-uvm_object.html" title="uvm_object"/>
  </item>
  <item src="docs/html/src/overviews/reporting.txt" title="Reporting"/>
  <item src="docs/html/src/overviews/recording.txt" title="Recording"/>
  <item src="docs/html/src/overviews/factory.txt" title="Factory"/>
  <item src="docs/html/src/overviews/phasing.txt" title="Phasing">
    <item src="docs/html/src/overviews/test-phasing.txt" title="User Defined Phasing"/>
  </item>
</menu>
```

Chapter 5. Comments Formatting

There are two formatting styles supported: *JavaDoc* & *NaturalDocs*.

You can use **HTML syntax** in *JavaDoc* comments.

5.1 JavaDoc

```
/**
 * <h1>Function Description</h1> using <b>HTML Tags</b> and {@literal <b> JavaDoc </b> }
 * <ul><li>HTML list element 1</li><li>HTML list element 2</li></ul>
 * For more details: {@link http://www.dvteclipse.com/documentation/sv/Export_HTML_Documentation.html DVT Documentation}
 *
 * @param slave_name - first param
 * @param min_addr - second param
 * @param max_addr - third param
 * @return min_addr
 *
 * @see get_type
 * @see build_phase
 *
 * @author Author's name
 * @version 1.0
 */
function void set_slave_address_map(string slave_name,
  int min_addr, int max_addr);
  ubus_slave_monitor tmp_slave_monitor;
  if( bus_monitor != null ) begin
    // Set slave address map for bus monitor
    bus_monitor.set_slave_configs(slave_name, min_addr, max_addr);
  end
  // Set slave address map for slave monitor
  $cast(tmp_slave_monitor, lookup({slave_name, ".monitor"}));
  tmp_slave_monitor.set_addr_range(min_addr, max_addr);
  return min_addr;
endfunction : set_slave_address_map
```

public void

set_slave_address_map(string slave_name, int min_addr, int max_addr)

Function Description

using **HTML Tags** and JavaDoc

- HTML list element 1
- HTML list element 2

For more details: DVT Documentation

Returns:
min_addr

Arguments:
slave_name - first param
min_addr - second param
max_addr - third param

See Also:
get_type
build_phase

Version:
1.0.

Author:
Author's name

The table below lists the JavaDoc tags that DVT recognizes. For more details see http://en.wikipedia.org/wiki/Javadoc.

@param	**Valid for: functions, tasks** Adds a parameter with the specified argument-name followed by the specified description to the *"Arguments"* section
@return	**Valid for: functions** Adds a *"Returns"* section with the description text. This text should describe the return type and permissible range of values
@see	Adds a *"See Also"* heading with a link or text entry that points to reference. A doc comment may contain any number of @see tags, all grouped under the same heading
@author	Adds an *"Author"* entry
@deprecated	Adds a comment indicating that this API should no longer be used (even though it may continue to work)
@version	Adds a *"Version"* subheading with the specified version-text
@since	Adds a *"Since"* heading with the specified since-text
{@link LINK_ADDRESS LINK_TEXT}	Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. This tag is valid in all doc comments. For more details see below.
{@literal text}	Displays text without interpreting the text as HTML markup or nested javadoc tags. This enables you to use regular angle brackets (< and >) instead of the HTML entities (<and >) in doc comments, such as in parameter types (<Object>), inequalities (3 < 4), or arrows (<-)

JavaDoc Links

An in-line link in a comment can be created using this tag **{@link LINK_ADDRESS LINK_TEXT}**.There are two types of links:

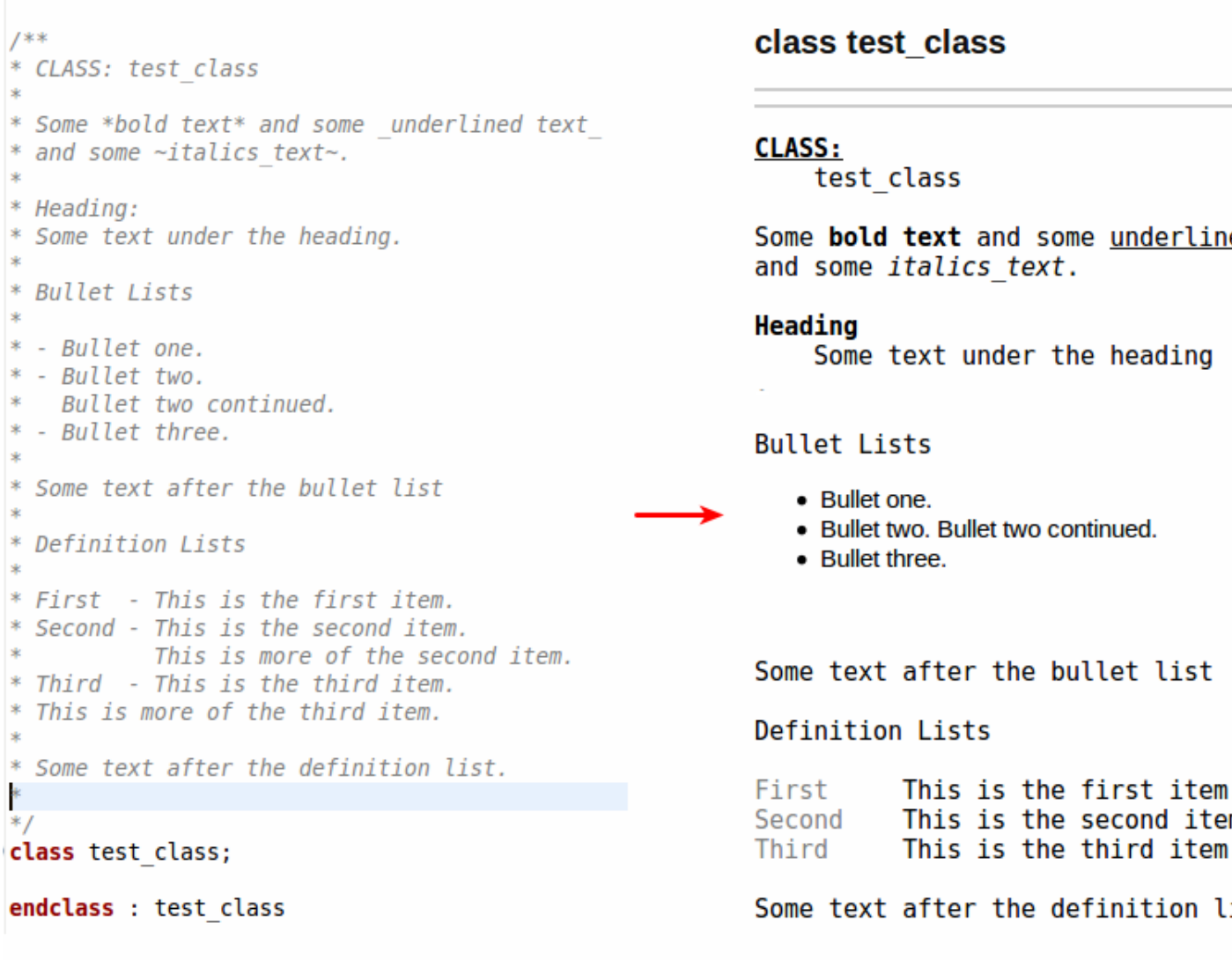
- **Internal Links** -> point to data inside Documentation. In this case **LINK_ADDRESS** must respect the following notation: **Package_Name::Class_Name.Method_Name** for an absolute path or

TYPE_NAME.INNER_TYPE_NAME or just **TYPE_NAME** for relative paths. In case of a relative path a link will be created to the best match for that type with regard to its scope inside the project.
NOTE: Using relative paths could generate broken links if there are different data types with the same name inside the project!

- **External Links** -> point to external web pages or files. For webpages **LinkAddress** must start with **http://** and for files with **file://** followed by the resource's address. For example: {@link https://www.dvteclipse.com} or {@link file://external-res.pdf}

For both types of links **LINK_TEXT** is optional and it can be used to show a user defined text instead of link's path.

5.2 NaturalDocs



The table below lists the NaturalDocs syntax that DVT recognizes. For more details see http://www.naturaldocs.org.

<LinkAddress>	Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. This tag is valid in all doc comments. For more details see below.
~Italic Text~	Use tilda (~) for *Italic Text*.
Bold Text	Use star (*) for **Bold Text**.
Headings	You can add headings to your output just by ending a line with a colon and having a blank line above it.
Bullet Lists	You can add bullet lists by starting a line with **a dash, an asterisk, an o, or a plus**. Bullets can have blank lines between them if you want, and subsequent lines don't have to be

	indented. You end a list by skipping a line and doing something else.
Definition Lists	You can add a definition list by using the format below, specifically “text space dash space text”. Like bullet lists, you can have blank lines between them if you want, subsequent lines don’t have to be indented, and you end the list by skipping a line and doing something else.
Images	You can include images in your documentation by writing *(see filename)*. If you put it alone on a line it will be embedded in place.
Code and Text Diagrams	, or :'". If you have a vertical line or text box with the comment, you must separate these symbols from it with a space.

NaturalDocs Links

An in-line link in a comment can be created using **<LINK_ADDRESS LINK_TEXT>**.There are two types of links:

- **Internal Links** -> point to data inside Documentation. In this case **LINK_ADDRESS** must respect the following notation: **Package_Name::Class_Name.Method_Name** for an absolute path or **TYPE_NAME.INNER_TYPE_NAME** or just **TYPE_NAME** for relative paths. In case of a relative path a link will be created to the best match for that type with regard to its scope inside the project.
  NOTE: Using relative paths could generate broken links if there are different data types with the same name inside the project!

- **External Links** -> point to external web pages or files. For webpages **LinkAddress** must start with **http://** and for files with **file://** followed by the resource's address. For example: < https://www.dvteclipse.com> or <file://external-res.pdf>

For both types of links **LINK_TEXT** is optional and it can be used to show a user defined text instead of link's path.

Chapter 6. Customizing Documentation

Generated documentation using new HTML style can be customized using the files **<html_doc>/css/custom.css** and **<html_doc>/js/custom.js**.

The new HTML style is based on Bootstrap [http://http://getbootstrap.com/] and jQuery [http://jquery.com/] frameworks and customizations can be done using jQuery API and by changing Bootstrap's default styles.

Custom CSS

The custom Cascading Style Sheet is included last and can overwrite any style defined above.

Example: changing the color of collapsible panels holding the element names.

```
.panel-default > .panel-heading {
   background-color: Chocolate;
   color: Cornsilk;
}
```

Custom JavaScript

The custom.js JavaScript file is included at the end of the HTML and it can completely change the content of the page.

Adding a custom "CONFIDENTIAL" warning to TOC and every page header can be done using jQuery.

```
$('body').prepend('<div class="alert alert-warning text-center" role="alert">CONFIDENTIAL</div>');
```

Customizing the TOC page and the content pages separately can be achieved by testing the window name:

```
if (window.name === "content")
   $('body').prepend('<div class="alert alert-warning text-center" role="alert">This is the content frame</div>

if (window.name === "toc")
   $('body').prepend('<div class="alert alert-warning text-center" role="alert">This is the toc frame</div>');
```

A similar approach can be used to customize individual pages based on the file name:

```
if (window.location.pathname.split('/').pop() === "summary-overview.html")
   $('body').prepend('<div class="alert alert-warning text-center" role="alert">This is the the summary page<
```

Customizing the "Generated from" links that point to the relative path of the file used to generate that page can be achieved by maching all the links with the href starting with *../sv* and replacing the href prefix with the new prefix *http://some/external/location/sv* :

```
if (window.name === "content") {
```

```
 $('hr')
 .next('a[href^="../sv"]')
 .attr('href', function() {
  return this.href.replace(/^file:.*\/sv/, 'http://some/external/location/sv')
 })
 .attr('target', function() {
  this.target = "_blank";
 });
}
```

Chapter 7. What is New?

#.# - major version - Includes new features, major enhancements, architectural changes, bug fixes.
Since 2015, a major version is named in sync with the release year, for example the first major version of 201
NOTE: When switching to a new major version it is recommended to start in a new workspace.

#.#.# - minor version - Includes bug fixes, minor enhancements.

17.1.7 (10 April 2017)

Bugfixes

- DVT-8800 Comment lines with words containing the element name are stripped from documentation
- DVT-9796 Specador: Go to element from global search does not work for mixed-language documentation

17.1.1 (24 February 2017)

Features

- DVT-3079 Generate Finite-State Machine Diagrams

Enhancements

- DVT-7673 Fail when files passed as arguments do not exist

16.1.37 (24 February 2017)

Bugfixes

- DVT-9375 Diagrams from referenced documentation are regenerated if the reference is outside of the project

16.1.35 (1 February 2017)

Enhancements

- DVT-9146 Add the diagram-max-nof-nodes in DTD for auto complete

Bugfixes

- DVT-9365 Global scope API filter does not work
- DVT-9418 Macros are documented even if excluded but Global Scope is selected

16.1.31 (9 December 2016)

Enhancements

- DVT-8159 Ignore @brief tags lines in comments

- DVT-9132 Add covergroup information in class, struct, unit pages
- DVT-9131 Ability to skip a header comment candidate that matches a simple pattern or regex when using +dvt_extract_comment_header+
- DVT-9134 Ignore invalid HTML tags when parsing comments as JavaDoc

Bugfixes

- DVT-9293 Fix NullPointerException when +dvt_auto_snps_vip_* flags are used

16.1.27 (28 October 2016)

Bugfixes

- DVT-9122 Build config: irun location is not correctly inferred when compiling in batch mode

16.1.22 (12 September 2016)

Bugfixes

- DVT-8948 When testbench classes reside under a program, they are not available in the main index

16.1.16 (8 July 2016)

Enhancements

- DVT-8135 Check that executed script is part of the same distribution where $DVT_HOME points to

16.1.9 (9 May 2016)

Features

- DVT-8567 Ability to add block, flow and schematic design diagrams using export-design-block-diagram, export-design-flow-diagram, export-design-schematic-diagram

Bugfixes

- DVT-7496 Expand on e Language checks page groups doesn't work

16.1.2 (3 March 2016)

Bugfixes

- DVT-8326 No documentation generated for inner enums, structs or classes
- DVT-8340 Wrong Java path in MacOS distros

16.1.1 (24 February 2016)

Enhancements

- DVT-7978 Updated JRE in distribution to version 1.8.0u66
- DVT-8275 Build with Java 8, minimal JRE required version increased to 1.8

15.1.37 (23 December 2015)

Enhancements

- DVT-8156 SystemVerilog: Add preference to enable/disable "Ifndef Guard Defines" extraction to HTML, do not generate by default
- DVT-8157 SystemVerilog: Add preference to enable/disable "Control Defines" extraction to HTML

15.1.34 (28 November 2015)

Enhancements

- DVT-8111 Add a preference to skip class diagram generation if maximum number of nodes exceeds a specified threshold

15.1.33 (20 November 2015)

Performance

- DVT-8092 SystemVerilog: Improve the performance of assertions and packages analysis

15.1.32 (18 November 2015)

Bugfixes

- DVT-8085 Sometimes license checkout fails when using the latest FlexLM server (11.13.1)

15.1.27 (8 October 2015)

Bugfixes

- DVT-7927 Wrong package comment processing when used for the overview page

15.1.25 (22 September 2015)

Enhancements

- DVT-7836 Ability to create URL for HTML frame content in order to simplify sharing links to specific pages

15.1.24 (18 September 2015)

Enhancements

- DVT-7832 Ability to use a specific package documentation as the overview page
- DVT-7833 Ability to filter UVM API from index and macros pages in order to avoid clutter

- DVT-7834 Show functions and tasks in generated documentation for interfaces
- DVT-7835 Provide legend for class diagrams in generated documentation

Bugfixes

- DVT-7888 Wrong progress report when linking external documentations

15.1.22 (2 September 2015)

Enhancements

- DVT-7829 Add diagram generation process timeout (1 minute)

15.1.18 (10 August 2015)

Performance

- DVT-7739 Improve Specator HTML search box performance

Enhancements

- DVT-7737 Add preference to show brief comment in index tables

15.1.17 (3 August 2015)

Features

- DVT-7727 New specador.bat Windows script

Enhancements

- DVT-6930 Ability to link header comment to first significant element in file

15.1.16 (27 July 2015)

Deprecated

- -gen_html_doc_from_settings is deprecated, use -preferences instead
- -get_html_doc flag is deprecated, use -title instead

Enhancements

- DVT-7664 Enhance progress reporting in batch mode - what file is currently generated, how long it takes
- DVT-7665 Use both extern and implementation function argument comments when generating documentation
- DVT-7667 Don't modify the capitalization of the first word in the sentence if that word is in fact the identifier name

- DVT-7669 Ability to pass custom menu by command line
- DVT-7670 Ability to pass title by command line

Bugfixes

- DVT-7663 Use portable awk syntax in scripts
- DVT-7687 Avoid silent exit after a StackOverflowError or OutOfMemoryError

15.1.11 (20 May 2015)

Bugfixes

- DVT-7474 License error due to a NullPointerException in FlexLM

15.1.10 (15 May 2015)

Bugfixes

- DVT-7449 RuntimeExceptions are thrown when generating documentation that contains some diagram types

15.1.1 (27 February 2015)

Enhancements

- DVT-7065 Build for Java 7, minimal JRE required version increased to 1.7

3.5.35 (30 January 2015)

Bugfixes

- DVT-6284 Diagrams in Specador should reflect architectures not entities in VHDL

3.5.32 (18 December 2014)

Bugfixes

- DVT-6900 Generated design diagrams are empty

3.5.30 (28 November 2014)

Bugfixes

- DVT-6854 No license found when using Specador for SystemVerilog

3.5.26 (31 October 2014)

Enhancements

- DVT-6766 Use new HTML look & feel by default

3.5.25 (23 October 2014)

Enhancements

- DVT-6431 VHDL: Added architecture instances and sub-instances

3.5.24 (10 October 2014)

Enhancements

- DVT-6278 Added compile waivers in functionality
- DVT-6710 Added support for clocking blocks

3.5.23 (2 October 2014)

Features

- DVT-6686 XML user defined menu - see XML Menu File Syntax

Enhancements

- DVT-6213 Refine parameters documentation
- DVT-6688 Show parameters in the inheritance tree
- DVT-6689 Refine interfaces documentation (ports, variables, modports, clocking blocks)

3.5.19 (21 August 2014)

Bugfixes

- DVT-6553 Specador compiles all files according to -lang switch in a mixed language build configuration regardless of extension mappings
- DVT-6565 Specador complains about non-existing irun executable even if not in ius.irun compatibility mode
- DVT-6570 FileNotFoundException (Not a directory) when generating documentation using the new HTML style

3.5.18 (1 August 2014)

Bugfixes

- DVT-6470 Specador "-h" shows help but also prints an error

3.5.17 (25 July 2014)

Features

- DVT-5560 Ability to customize the HTML look & feel when new HTML style is used

- DVT-6487 Ability to inject HTML in generated documentation when new HTML style is used

Bugfixes

- DVT-6526 Watermark footer missing in new HTML style

3.5.16 (8 July 2014)

Features

- DVT-6485 Specador: New HTML look & feel

Enhancements

- DVT-6279 SystemVerilog API defined under a class should appear only under that class and not under global API
- DVT-6434 SystemVerilog remove covergroups, assertions, functions, tasks, variables from index page and search

Bugfixes

- DVT-5550 Specador: Method argument comments are not extracted
- DVT-6210 Specador: Fixed JavaDoc @link hyperlink extraction when similar links are used

3.5.14 (24 June 2014)

Bugfixes

- DVT-6353 VHDL Overloaded functions are not visible

3.5.13 (13 June 2014)

Enhancements

- DVT-6282 Validate the settings XML before compilation

Bugfixes

- DVT-6331 Wrong hyperlinks when including external documentation directories
- DVT-6333 Show the external documentation title under the 'Referenced Documentation' section in TOC

3.5.12 (10 June 2014)

Bugfixes

- DVT-6285 & DVT-6287 Exceptions when generating documentation in certain configurations

- DVT-6286 Exception when generating documentation with module diagrams with ports
- DVT-6288 FileNotFoundException when documentation is generated with diagrams in the same location for multiple projects
- DVT-6304 Progress dialog should also include design diagrams and design diagrams with ports

3.5.11 (30 May 2014)

- First version

Chapter 8. Legal Notices

Chapter 9. Third Party Licenses

The following software may be included in this product:

- **Eclipse Platform and Eclipse Plugins** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_ECLIPSE.TXT])
- **Java SE Runtime Environment (JRE)** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_JRE.TXT])
- **ANTLR v2** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_ANTLR.TXT])
- **FreeMarker** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_FREEMARKER.TXT])
- **Graphviz** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_GRAPHVIZ.TXT])
- **PMD** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_PMD.TXT])
- **P4Eclipse** (see license [https://www.dvteclipse.com/third_party_licenses/DISCLAIMER_P4ECLIPSE.TXT])
- **SVNKit** (see license [https://www.dvteclipse.com/third_party_licenses/LICENSE_SVNKIT.TXT])
- **viPlugin** (DVT includes a version of Michael Bartl's viPlugin)