

# Collecting Doxygen Comments

## Abstract

This document describes how to collect Doxygen code comments as specification for modules and test objects. The Doxygen comments will automatically be collected or updated when analyzing a module. This feature requires Doxygen being installed on the computer where the code comments shall be collected. See <http://doxygen.org> for reference.

## Table of Contents

Abstract .....	1
1 Introduction.....	2
2 Configuration .....	2
3 Example .....	3

# 1 Introduction

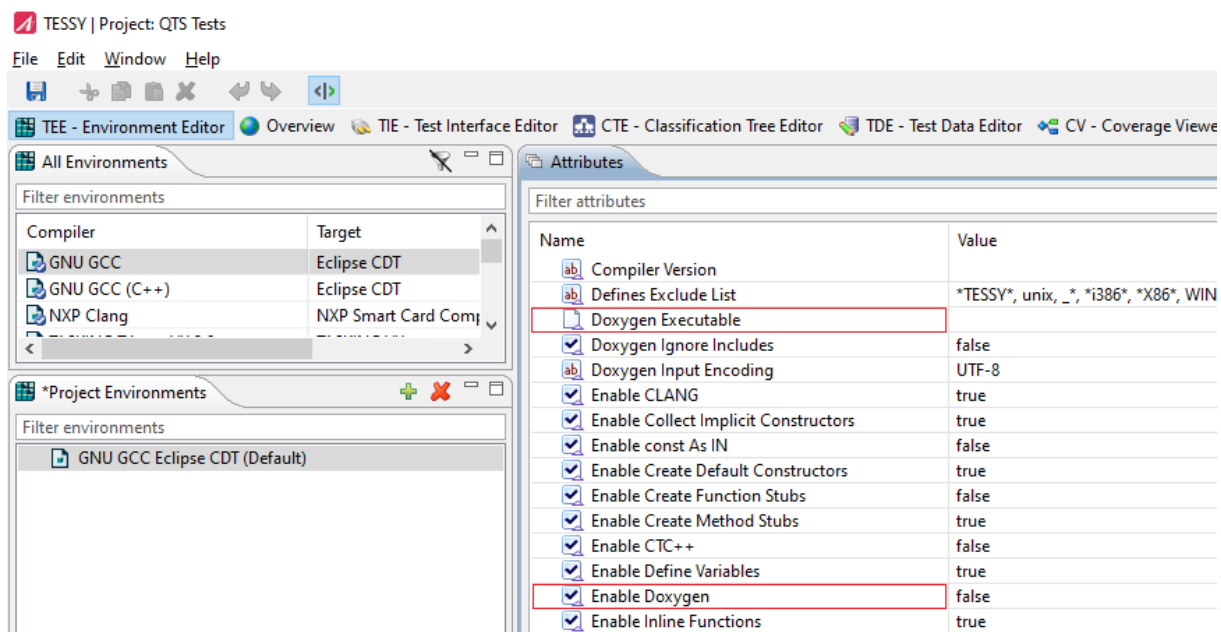
Doxygen provides means to generate a HTML documentation from specifically formatted code comments. These comments can be parsed by TESSY and assigned to modules and test objects as specification text. As a result, these HTML formatted code comments will appear within the TESSY test reports.

# 2 Configuration

The following attributes need to be set in order to activate the Doxygen feature:

- Enable Doxygen                      *Set to true to run Doxygen after module analysis*
- Doxygen Executable                *Path to doxygen.exe on your system*

The attribute can be set individually on module level as well as within the TEE environment configuration which would be the suggested way to have this feature available for a whole project.



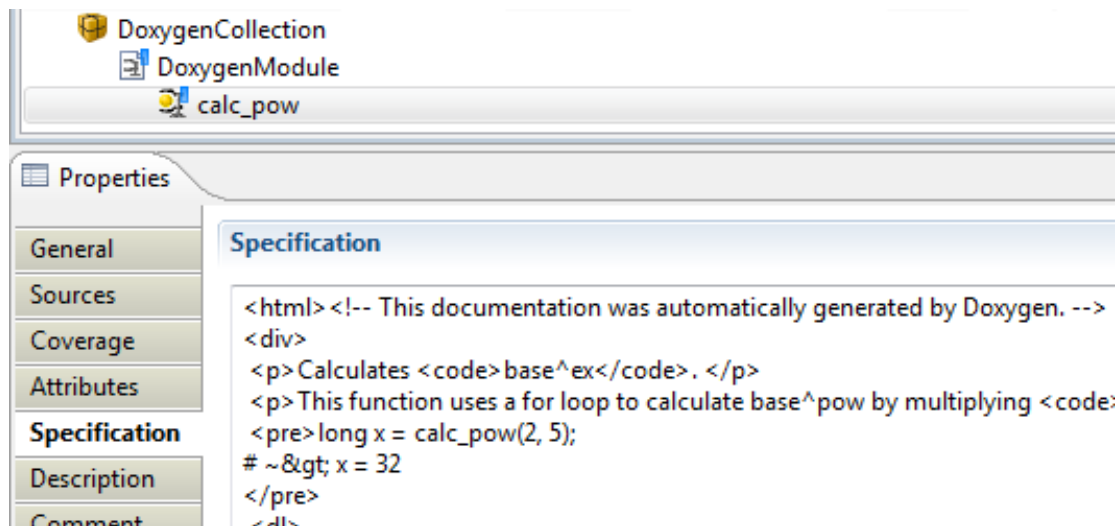
### 3 Example

First of all, you need to document your C/C++ code and with Doxygen comments, e.g. like this:

```
/**
 * @brief Calculates @c base^ex
 *
 * This function uses a for loop to calculate base^ex
 * by multiplying @p base @p ex -times with itself.
 * Example Usage:
 * @verbatim
 * long x = calc_pow(2, 5);
 * # ~> x = 32
 * @endverbatim
 *
 * @warning The function does not validate its input
 */
long calc_pow(long base, long ex) {
    ...
}
```

These comments can be either be written in the header or source files wherever preferred. If the project is configured correctly (See chapter 2) and the module gets analyzed again, the “Specification” property of each function should automatically get filled in with the HTML string generated by Doxygen. If any errors occur, a message gets displayed in the console explaining the problem. An undocumented function will keep a blank specification and custom specifications will not get overwritten by automatically generated ones.

The previous example would look like this inside TESSY after module analysis:



Doxygen also allows file documentation comments providing details about the file content:

```
/**
 * @file example.c
 * @author My Self
 * @brief Example file
 */
```

These will also be collected from all linked source files and written into the module specification.

In a generated report the specifications strings will get rendered as actual HTML:

Comments/Description/Specification	
Name	Text
Module 'DoxygenModule'	<i>example.c:</i> Example file. Author My Self
Test Object 'calc_pow'	Calculates base <sup>ex</sup> . This function uses a for loop to calculate base <sup>ex</sup> ... <pre>long x = calc_pow(2, 5); # ~&gt; x = 32</pre> Warning The function does not validate its input

If a module has multiple source files with file documentation comments, they will get displayed in a list:

Comments/Description/Specification	
Name	Text
Module 'ManyFiles'	<i>file1.c:</i> Sample file 1.
	<hr/> <i>file2.c:</i> Sample file 2.
	<hr/> <i>file3.c:</i> Sample file 3.
	<hr/> <i>file4.c:</i> Sample file 4.