

# T3D Technical Documentation

This documentation serves as a technical and development reference for maintenance and further development.

## Functioning

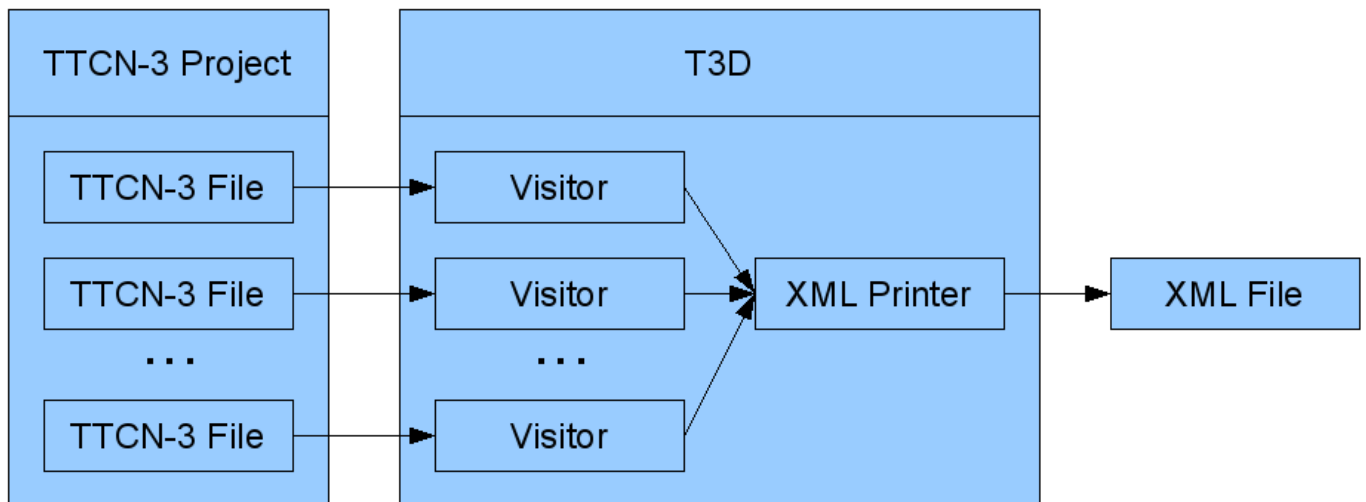
The HTML documentation is created in 2 steps:



First, the TTCN-3 files are parsed and the information required for step two is written into XML files. These files are then used to generate the HTML files via XSL-Stylesheets. The XSL-Transformations are done by the Saxon XSLT Processor.

## Creating the XML Files

TTCN-3 trees are generated by TRex and are then traversed by **Visitors** which gather all relevant information to be written into the XML files. The **Visitors** are instantiated for every TTCN-3 file:



## Visitors

There are currently 3 Visitors:

- **T3DVisior** - gathers information used to generate the **project.xml**
- **DependencyVisitor** - gathers information used to generate the **dependencies.xml**
- **ImportVisitor** - gathers information used to generate the **imports.xml**

### Creating new Visitors

To create a new Visitor to implement new functionalities or to gather new information from the TTCN-3 files, the following code can be used.

*Basic structure of the new Visitor:*

```
package org.etsi.t3d.visitor;

import de.ugoe.cs.swe.trex.core.analyzer.rfparser.LocationAST;

public class NewVisitor extends AbstractVisitor {
    @Override
    public void finish() {
    }

    @Override
    public void init() {
    }

    /*
        visitNodeType(LocationAST node) functions
    */
}
```

*visitNodeType(LocationAST node)* functions are called when nodes of certain types are traversed. For example, *visitFunctionDef(LocationAST node)* is called when a *FunctionDef* node is found. This node is represented by the *node* parameter. See *AbstractVisitor.java* for a full list of those functions.

### Adding the new Visitor to T3D.java

The following code shows where to implement the new Visitor in T3D.java, so it will function properly.

*T3D.java:*

```
/* ... */
public void run(String[] args) {
    /* ... */
    for (int i = 0; i < ttcn3Resources.size(); i++) {
        analyzer = analyzerFactory.getTTCN3Analyzer(ttcn3Resources.get(i));
        T3DVisitor visitor = new T3DVisitor(xmlPrinter);
        DependencyVisitor dependencyVisitor = new DependencyVisitor(depPrinter);
        ImportVisitor importVisitor = new ImportVisitor(importPrinter);

        NewVisitor newVisitor = new NewVisitor();
        // ...
        newVisitor.acceptDFS((LocationAST) analyzer.getParser().getAST());
        // ...
    }
    /* ... */
}
/* ... */
```

## Structure of the generated XML files

### project.xml

This file contains the most comprehensive information and represents an abstracted version of the source TTCN-3 files, including module structure, the individual elements, their bodies (if configured so), and T3Doc comments for every module, group, and element of the processed TTCN-3 files. It serves as a basis for the main and the module parameter / test case views.

DTD:

```
<!ELEMENT project (module, element, group)*>
<!ELEMENT module (name, comment?, behaviour, modulename, import*)>
<!ELEMENT group (name, location, comment?, behaviour, modulename, path)>
<!ELEMENT element (name, location, comment?, behaviour, modulename, mpview_ModuleParDef?, path)
  <!ATTLIST element type CDATA #REQUIRED>

<!ELEMENT path (path_group)>
<!ELEMENT path_group (path_group)>
  <!ATTLIST path_group loc CDATA #REQUIRED name CDATA #REQUIRED>
<!--
Contains information on which groups an element or group belongs to
<path>
```

```

<path_group loc="..." name="subsubgroup">
  <path_group loc="..." name="subgroup">
    <path_group loc="..." name="group">
      </path_group>
    </path_group>
  </path_group>
</path>
-->

<!ELEMENT mpview_ModuleParDef((mpview_TestcaseDef | mpview_FunctionDef | mpview_AltstepDef)*)
<!ELEMENT mpview_FunctionDef((mpview_TestcaseDef | mpview_FunctionDef | mpview_AltstepDef)*)
<!ELEMENT mpview_TestcaseDef((mpview_TestcaseDef | mpview_FunctionDef | mpview_AltstepDef)*)
<!ELEMENT mpview_AltstepDef((mpview_TestcaseDef | mpview_FunctionDef | mpview_AltstepDef)*)
  <!ATTLIST mpview_ModuleParDef loc CDATA #REQUIRED name CDATA #REQUIRED>
  <!ATTLIST mpview_FunctionDef loc CDATA #REQUIRED name CDATA #REQUIRED>
  <!ATTLIST mpview_TestcaseDef loc CDATA #REQUIRED name CDATA #REQUIRED>
  <!ATTLIST mpview_AltstepDef loc CDATA #REQUIRED name CDATA #REQUIRED>
<!-- contains information on how module parameters are connected to testcases -->

<!ELEMENT behaviour (#PCDATA | keyword | constructbody | link)*>
<!ELEMENT constructbody (#PCDATA | keyword | link)*>
  <!ATTLIST constructbody id CDATA #REQUIRED>
<!ELEMENT keyword (#PCDATA)>
<!-- <keyword> elements are used around keywords of TTCN3 like "function" -->
<!-- <link> elements are used to identify crossreferences -->
<!-- the <constructbody> tag surrounds the parts of the behaviour which can be toggled in the

<!ELEMENT link (#PCDATA)>
  <!ATTLIST link loc CDATA #REQUIRED>

<!ELEMENT newline EMPTY>
<!ELEMENT tab EMPTY>

<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>

<!ELEMENT comment (author | config | desc | exception | member | param | purpose | remark
                  | return | see | since | status | url | verdict | version)*>
  <!-- @version 2.0.1 ist stored as <version>2.0.1</version> -->

<!ELEMENT author (#PCDATA | esee | eurl)*>
<!ELEMENT config (#PCDATA | esee | eurl)*>
<!ELEMENT desc (#PCDATA | esee | eurl)*>
<!ELEMENT exception (#PCDATA | esee | eurl)*>
<!ELEMENT purpose (#PCDATA | esee | eurl)*>
<!ELEMENT param (#PCDATA | esee | eurl)*>
<!ELEMENT remark (#PCDATA | esee | eurl)*>
<!ELEMENT return (#PCDATA | esee | eurl)*>

```

```
<!ELEMENT see (#PCDATA)>
<!ELEMENT since (#PCDATA | esee | eurl)*>
<!ELEMENT status (#PCDATA | esee | eurl)*>
<!ELEMENT url (#PCDATA | esee | eurl)*>
<!ELEMENT verdict (#PCDATA | esee | eurl)*>
<!ELEMENT version (#PCDATA | esee | eurl)*>
<!ELEMENT member (#PCDATA | esee | eurl)*>

<!ELEMENT esee (#PCDATA)>
<!ELEMENT eurl (#PCDATA)>
```

Examples

	TTCN-3	
Behaviour	<pre>testcase tc2() runs on sampleComponent1 system sampleComponent3{     map(mtc:p1, system:p1)     var sampleComponent2 component2, component3;     component2 := sampleComponent2.create;     component3 := sampleComponent2.create;     component2.start(f3());     component3.start(f4(1, "a"));     unmap(mtc:p1, system:p1) }</pre>	<pre>&lt;behaviour&gt;&lt;keyword&gt;     &lt;keyword&gt;runs&lt;/     &lt;keyword&gt;system     &lt;keyword&gt;map&lt;     &lt;keyword&gt;var&lt;     component2 :=     component3 :=     component2.st     component3.st     &lt;keyword&gt;unma }&lt;/constructbody&gt; &lt;/behaviour&gt;</pre>
Path	<pre>group group1{     group group1_1{         group group1_1_1{             //...         }     } }</pre>	<pre>&lt;path&gt; &lt;path_group loc="gr &lt;path_group loc="g &lt;path_group loc=" &lt;/path_group&gt; &lt;/path_group&gt; &lt;/path_group&gt; &lt;/path&gt;</pre>

import.xml

This file contains information about the imports and dependency relations of the processed TTCN-3 modules. It serves as a basis for the import / module dependency view.

DTD:

```

<!ELEMENT imports module*>
<!ELEMENT module import*>
<!ELEMENT import (import_behaviour, import*) | EMPTY>
  <!ATTLIST import name CDATA #REQUIRED>
<!ELEMENT import_behaviour (#PCDATA | link)*>
<!ELEMENT link (#PCDATA)>
  <!ATTLIST link loc CDATA #REQUIRED>

```

## dependencies.xml

This file can be thought of as a blend between an abstracted version of the main project.xml file and a low-level version of the import.xml file, featuring a compact representation of the low-level dependencies at the module definition (element) level - it contains all the module definitions and all the known elements referenced directly within each module definition. There is no separate view associated to this file, since it is basically a compact representation of the main view and its main intent is the use with third-party tools to perform custom tasks, such as slicing or markup of definitions related to a particular module definition (e.g. approved / locked definitions, etc.).

### DTD:

```

<!ELEMENT dependencies element*>
<!ELEMENT element refflist>
  <!ATTLIST element id ID #REQUIRED,
                    type CDATA #REQUIRED
                    line CDATA #REQUIRED
                    module CDATA #REQUIRED>
<!ELEMENT refflist ref*>
<!ELEMENT ref EMPTY>
  <!ATTLIST ref id IDREF #REQUIRED>

```

## log.xml

This file contains all informations, warnings and errors for each TTCN-3 module that are logged by T3D.

### DTD:

```

<!ELEMENT t3dlog file*>
  <!ATTLIST t3dlog t3dversion CDATA #REQUIRED>

  <!ELEMENT file (warning | error )*>
  <!ATTLIST file path CDATA #REQUIRED>

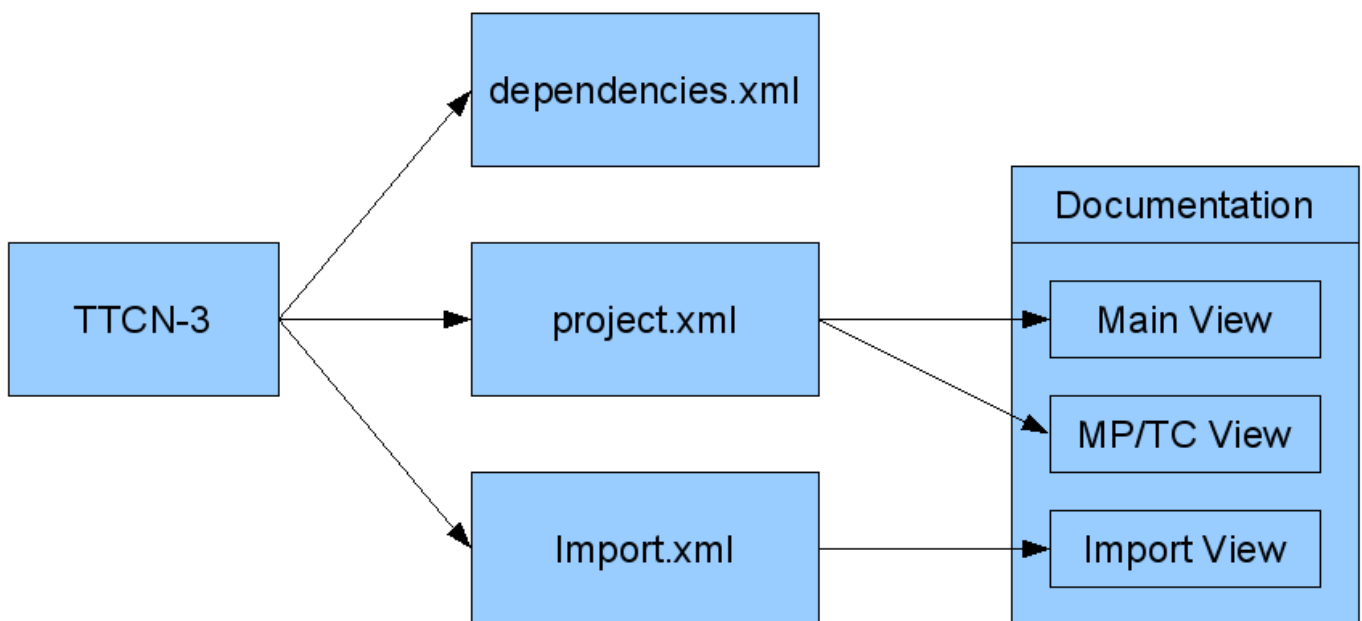
```

```
<!ELEMENT warning #PCDATA>
  <!ATTLIST warning level CDATA #REQUIRED
                lines CDATA #REQUIRED>

<!ELEMENT error #PCDATA>
  <!ATTLIST error level CDATA #REQUIRED
                lines CDATA #REQUIRED>
```

## Generation of HTML files

The HTML documentation is generated with the **project.xml** and the **imports.xml**:



The **Main View** and **Module Parameter/Testcase? View** of the documentation are generated with the **html.xsl** stylesheet using the **project.xml** and the **Import View** is generated with the **html\_import.xsl** stylesheet using the **imports.xml** .

## Structure of the XSL-stylesheets

Below, the structure and templates of the XSL-Stylesheets are described to simplify making modifications. See [{\\$T3D\\_HOME}/css/doc.css](#) for descriptions of the HTML elements.

### html.xsl

Structure of the html.xsl:

```
<xsl:stylesheet>
  <xsl:template match="/">
    <xsl:result-document href="..." format="html">
      <!-- generates index.html -->
    </xsl:result-document>

    <xsl:result-document href="..." format="html">
      <!-- generates import_index.html -->
    </xsl:result-document>

    <xsl:result-document href="..." format="html">
      <!-- generates mp_index.html -->
    </xsl:result-document>

    <xsl:for-each select="//module">
      <xsl:variable name="currentmodule" select="name/text()" />
      <xsl:variable name="currentindex">
        <!--...-->
      </xsl:variable>
      <xsl:variable name="currentmp_index">
        <!--...-->
      </xsl:variable>

      <xsl:result-document href="..." format="html">
        <!-- generates Main View Documentation of modules -->
      </xsl:result-document>

      <xsl:result-document href="..." format="html">
        <!-- generates MP/TC View Documentation of modules -->
      </xsl:result-document>

      <xsl:for-each select="//element[modulename/text() eq $currentmodule]">
```



```

<xsl:result-document href="..." format="html">
  <!-- generates Main View Documentation of elements -->
</xsl:result-document>

<xsl:if test="@type eq 'testcase'">
  <xsl:result-document href="..." format="html">
    <!-- generates MP/TC View Documentation of testcases -->
  </xsl:result-document>
</xsl:if>

<xsl:if test="@type eq 'parameter'">
  <xsl:result-document href="..." format="html">
    <!-- generates MP/TC View Documentation of module parameters -->
  </xsl:result-document>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="//group">
  <xsl:result-document href="..." format="html">
    <!-- generates Main View Documentation of groups -->
  </xsl:result-document>

  <xsl:result-document href="..." format="html">
    <!-- generates MP/TC View Documentation of groups -->
  </xsl:result-document>
</xsl:for-each>
</xsl:template>

<!--
  Templates
-->

</xsl:stylesheet

```

The following table contains descriptions of parameters and function of the used XSLT templates in **html.xsl** .

Template	Parameters	Function
<b>main_allElements</b>	<b>\$this</b> = module grouplelement <b>\$currentindex</b> = complete construct index	calls <b>views</b> template inserts <b>\$currentindex</b> calls <b>header</b> template calls <b>element_withcomment</b> template
<b>mp_allTables</b>		

Template	Parameters	Function
	<b>\$testcases</b> = testcases <b>\$parameters</b> = parameters	generates <b>div_mp</b> element for groups, modules and the MP/TC index
<b>html_head</b>	<b>\$title</b> = title of page	generates <b>head</b> element
<b>views</b>	<b>\$main</b> = location of Main View documentation of the current element <b>\$mp</b> = location of MP/TC View documentation of the current element <b>\$import</b> = location of Import View documentation of the current element <b>\$togglemode</b>	generates <b>div_views</b> element and <b>T3D Generation Stamp</b>
<b>main_index</b>	<b>\$modulename</b> = name of the current module	generates <b>div_modules</b> element generates <b>div_index</b> element of the Main View of modules, elements and the index
<b>main_groupindex</b>	<b>\$grouplocation</b> = location of the current group	generates <b>div_modules</b> element generates <b>div_index</b> of the Main View of groups
<b>index_grouplist</b>	<b>\$this</b> = groups <b>\$prefix</b> = " or 'mp_'	generates list of hyperlinks to all groups given in parameter <b>\$this</b>
<b>index_elementlist</b>	<b>\$this</b> = elements <b>\$prefix</b> = " or 'mp_' <b>\$type</b> = type of elements	generates list of hyperlinks to all elements given in parameter <b>\$this</b>
<b>mp_index</b>	<b>\$modulename</b> = name of the current module	generates <b>div_modules</b> element generates <b>div_index</b> of the MP/TC View
<b>index_modulelist</b>	<b>\$prefix</b> = ", 'mp_' or 'import_'	generates list of hyperlinks to all modules
<b>header</b>	<b>\$this</b> = module grouplelement <b>\$prefix</b> = ", 'mp_' or 'import_'	generates <b>p_header</b> element
<b>element_withcomment</b>	<b>\$this</b> = module grouplelement	generates <b>div_content</b> element generates <b>div_comment</b> element generates <b>div_element</b> element
<b>path</b>	<b>\$this</b> = module grouplelement <b>\$prefix</b> = ", 'mp_' or 'import_'	calls <b>path_group</b> template
<b>path_group</b>	<b>\$this</b> = module grouplelement <b>\$prefix</b> = ", 'mp_' or 'import_'	generates the group part of the path navigation
<b>mp_testcase_table</b>	<b>\$this</b> = testcases	

Template	Parameters	Function
		generates the MP/TC View table of testcase elements
<b>mpview_path_testcase</b>	<b>\$this</b> = mpview_TestcaseDefs <b>\$first</b> = true/false	generates <b>Path</b> part of the MP/TC View table of testcase elements
match <b>mpview_ModuleParDef</b>		generates the MP/TC View table of module parameter elements
<b>mpview_path_modulepar</b>	<b>\$this</b> = moduleIgroupElement <b>\$first</b> = true/false	generates <b>Path</b> part of the MP/TC View table of module parameter elements

html\_import.xsl

The following table contains descriptions of parameters and function of the used XSLT templates in **html\_import.xsl** .

Template	Parameters	Function
<b>import</b>	<b>\$module</b> = name of the current module <b>\$importname</b> = name of imported module	generates a list of imports by <b>\$module</b> from <b>\$importname</b>
<b>importby</b>	<b>\$imports</b> = list of indirect imports	generates a list of <b>\$imports</b>
<b>show_imports</b>	<b>\$module</b> = module <b>\$currentmodule</b> = currently selected module	selects a color for <b>\$currentmodule</b> and calls the <b>colored_module</b> template
<b>colored_module</b>	<b>\$module</b> = module <b>\$color</b> = class of <b>\$module</b>	generates a colored hyperlink to <b>\$currentmodule</b> (used for list of modules in the middle column)

**html\_head****\$title** = title of page generates **head** element **legend** generates Import View Legend

## Using new XSL-Stylesheets and/or XML files

New XSL-Transformations can be executed by using the following code.

*XSL-Transformation:*

```
String foldername = getProperOutputPath() + "/" + getSubPath();
String xmlFileName = "name.xml";
String xsltFileName = "name.xsl";
String outputFileName = "outputName";
```

```

System.setProperty("javax.xml.transform.TransformerFactory", "net.sf.saxon.TransformerFactoryImpl");
TransformerFactory tfactory = TransformerFactory.newInstance();
Transformer transformer;
try {
    transformer = tfactory.newTransformer(new StreamSource(new File(xsltFileName)));
    //transformer.setParameter("parameterName", <parameterValue>);

    transformer.transform(new StreamSource(new File(foldername + "/" + xmlFileName)),
        new StreamResult(new FileOutputStream(foldername + "/" + outputFileNam
} catch (TransformerConfigurationException e) {
    e.printStackTrace();
} catch (TransformerException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
}

```

## Javascript

To toggle the visibility of certain elements in the HTML documentation (e.g. navigation items), a Javascript file called *doc.js* is used. It can be found in `{T3D_HOME}/js`.

## Functions

The Javascript file consists of the following functions:

- **getElementsByClass(searchClass,node,tag)** : returns all HTML elements of a certain class
- **toggleConstructbodies(hide)** : toggles visibility of all construct bodies in the Main View of elements/groups/modules if `hide == true`
- **toggleImportDetails()** : toggles visibility of all details in the Import View of a module
- **togglePaths(hide)** : toggles visibility of all Path columns in the MP/TC View of module parameters/test cases/groups/modules if `hide == true`
- **toggleHideNotes()** : toggles visibility of all (*toggle*) notes in the Main View of elements/groups/modules
- **toggleElement(element)** : toggles the visibility of *element*
- **toggle(elementName)** : calls **toggleElement()** with the element with the Id *elementName*
- **mp\_init(listName)** : collapses all construct lists in the construct index of the MP/TC View, except *listName*
- **init(listName)** : calls **toggleHideNotes()** and collapses all construct lists in the construct index of the Main View, except *listName*