ETSI TS 103 255 V1.1.1 (2015-03)

Methods for Testing and Specification (MTS);
TTCN-3 Conformance Test Suite for use of XML schema;
Abstract Test Suite (ATS) and
Implementation eXtra Information for Testing (IXIT)

Reference
DTS/MTS-103255

Keywords
conformance, testing, TSS&TP, TTCN, XML

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies the Abstract Test Suite (ATS) for the conformance test suite for using XML Schema with TTCN-3, as defined in ETSI ES 201 873-9 [1].

The objective of the present document is to provide a basis for conformance tests for TTCN-3 tools supporting "Using XML Schema with TTCN-3" extension [1]. The conformance test suite should give a high probability of standard conformance with respect to TTCN-3 tools from different vendors. In the present document only using XML Schema with TTCN-3, specified in ETSI ES 201 873-9 [1] have been considered but not the core language [10], tool implementation (see [i.1] and [i.2]), language mapping (see [i.3] and [i.4]) and language extension (see e.g. [i.5], [i.6] and [i.7]) aspects. The test notation used in the ATS attached in a zipped file is in TTCN-3 and it is part of the present document.

Annex A provides the Tree and Tabular Combined Notation (TTCN-3) part of the ATS.

Annex B provides the Partial Implementation Extra Information for Testing (PIXIT) Proforma of the ATS.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI ES 201 873-9 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 9: Using XML Schema with TTCN-3 ".

[2] ETSI ES 201 873-10 (V4.4.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 10: TTCN-3 Documentation Comment Specification ".

[3] ETSI TS 102 351: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); IPv6 Testing: Methodology and Framework".

[4] ISO/IEC 9646-1 (1992): "Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1: General concepts".

[5] ISO/IEC 9646-4: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 4: Test realization".

[6] ISO/IEC 9646-5: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 5: Requirements on test laboratories and clients for the conformance assessment process".

[7] ISO/IEC 9646-7 (1994): "Conformance testing methodology and framework - Part 7: Implementation Conformance Statement".

[8] ETSI TS 102 995: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Proforma for TTCN-3 reference test suite".

[9] ETSI TS 103 253: "Methods for Testing and Specification (MTS); TTCN-3 Conformance Test Suite for use of XML schema; Implementation Conformance Statement".

[10] ETSI ES 201 873-1 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI ES 201 873-5: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".

[i.2] ETSI ES 201 873-6: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

[i.3] ETSI ES 201 873-7: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 7: Using ASN.1 with TTCN-3".

[i.4] ETSI ES 201 873-8: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 8: The IDL to TTCN-3 Mapping". .

[i.5] ETSI ES 202 781: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Configuration and Deployment Support".

[i.6] ETSI ES 202 784: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Advanced Parameterization".

[i.7] ETSI ES 202 785: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Behaviour Types".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ISO/IEC 9646-1 [4], ISO/IEC 9646-7 [7], ETSI ES 201 873-1 [10] (TTCN-3), ETSI TS 102 995 [8] and the following apply:

Abstract Test Method (ATM): description of how an IUT is to be tested, given at an appropriate level of abstraction to make the description independent of any particular realization of a Means of Testing, but with enough detail to enable abstract test cases to be specified for this method

Abstract Test Suite (ATS): test suite composed of abstract test cases

ICS proforma: document, in the form of a questionnaire, which when completed for an implementation or system becomes an ICS

Implementation Conformance Statement (ICS): statement made by the supplier of an implementation claimed to conform to a given specification, stating which capabilities have been implemented

Implementation extra Information for Testing (IXIT): statement made by a supplier or implementor of an IUT which contains or references all of the information related to the IUT and its testing environment, which will enable the test laboratory to run an appropriate test suite against the IUT

Implementation Under Test (IUT): implementation of one or more OSI protocols in an adjacent user/provider relationship, being part of a real open system which is to be studied by testing

IXIT proforma: document, in the form of a questionnaire, which when completed for the IUT becomes the IXIT

Means Of Testing (MOT): combination of equipment and procedures that can perform the derivation, selection, parameterization and execution of test cases, in conformance with a reference standardized ATS and can produce a conformance log

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASCII	American Standard Code for Information Interchange
ATM	Abstract Test Method
ATS	Abstract Test Suite
BNF	Backus Nauru Form
CDATA	Character Data
ETS	Executable Test Suite
ICS	Implementation Conformance Statement
IUT	Implementation under Test
IXIT	Implementation extra Information for Testing
MOT	Means Of Testing
PIXIT	Partial Implementation Extra Information for Testing
TC	Test Case
TCI	TTCN-3 Control Interface
TP	Test Purpose
TRI	TTCN-3 Runtime Interface
TS	Test System
TSS	Test Suite Structure
TSS&TP	Test Suite Structure and Test Purposes
TTCN-3	Testing and Test Control Notation edition 3
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Abstract Test Method (ATM)

This clause describes the ATM used to test the conformance of TTCN-3 tool implementations as described in ETSI ES 201 873-9 "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 9: Using XML Schema with TTCN-3" [1]. Hereafter this standard will be referred to as TTCN-XML.

In the ATM, the work is performed on two levels:

- The TTCN-3 tool level. In TTCN-XML conformance tests, it is the TTCN-3 tool which is under test, i.e. the IUT. However, unlike in protocol conformance testing, it is not standardized how test inputs, i.e. TTCN-3 modules and XML Schema, are provided. Neither are there any standardized interfaces to monitor the reaction of the TTCN-3 tool to the test input. Outputs can only be observed indirectly by monitoring tool outputs such as tool specific command line information, graphical user interfaces, or test execution logs. The tool output is processed further in the tool output evaluation level in order to derive the tool conformance verdicts.

- The TTCN-3 tool output evaluation level. Here, the output of a TTCN-3 tool is indirectly observed, e.g. rejection of TTCN-3 code due to a compile-time error in a command line notification, logging of one or multiple test verdicts in a tool specific window or an execution trace. The observation is evaluated to assess the tool conformance as a result of stimulating the tool with the TTCN-3 modules. Compliance or support of the logging interface specified as part of the TTCN-3 Control Interface standard (TCI) is not required.

NOTE: The loading of the TTCN-3 modules and presentation of the output by the TTCN-3 tools is beyond the scope of the present document.

The ATS document contains the test inputs, i.e. TTCN-3 modules and XML Schemas, for TTCN-3 tools do not automate the execution of TTCN-3 tool conformance tests. TTCN-3 tool conformance test decisions shall be made on the basis of expected outputs as specified in the test purposes provided in the documentation and as part of the documentation of TTCN-3 tests in the ATS. Three different tool output classifications for TTCN-3 inputs exist:

- Rejection as invalid, i.e. the TTCN-3 input is declared syntactically or semantically incorrect by the tool. This can either happen at compile-time or at runtime.

- Rejection to execute, i.e. an ETS is produced from the test input, but an execution does not take place.

- Execution with results, i.e. the compiled or interpreted TTCN-3 code is executed and different kinds of outputs are produced that can be subject of an evaluation, for example, a logged TTCN-3 test verdict in a test execution trace (none, pass, fail, inconc) in a file or the console output. The respective tool is considered conformant to the test if the tool outputs match the expected execution results.

A conformance test for XML-supporting TTCN-3 tool can attempt to trigger every kind of such outputs in a controlled way, i.e. a test input that is rejected as invalid does not imply a failing conformance test verdict, but instead results in a pass verdict for the conformance test if the test is designed to trigger the rejection. More generally: a TTCN-3 tool conformance test passes if the tool output corresponds to the expected output. The range of expected outputs is described by the tool output classification above.

For a detailed description on how test verdict and test purposes are encoded and how they shall be evaluated with the ATS of annex A, please refer to clause 5.3.1.3 and the descriptions for the document tags @verdict and @purpose.

5 The ATS development process

5.1 Requirements and test purposes

For each test purpose there is a table defined in clause A.2 of ETSI TS 103 253 [9]. The requirements applicable to this TP are given by a reference to ETSI ES 201 873-9 [1]. There are no explicit formulations of requirements.

5.2 ATS structure

5.2.1 ATS folder tree

The ATS is split into folders, where each folder represents a clause in [1]. Clauses on a lower scope or hierarchy are mapped into subfolders. The names of the folders are derived from the clause names in the following way:

1) All clause and subclause numbers are converted to a two digit format: if the number consists of a single digit, it is prefixed with zero.

2) All spaces and dots in the clause number are removed and all digits are concatenated.

3) The clause name is transformed by converting all upper case letters to lower case and replacing spaces with low lines.

4) The transformed clause number and clause name are concatenated with a low line character inserted between them.

5) If a clause contains subclauses and there are also a test cases defined for the requirements defined in this clause, a special subfolder is created to accommodate these test cases. The folder name consists of a transformed clause number according to the above specified rules and the string "_top_level".

EXAMPLE: Clause 5 "Mapping XML schemas" of [1] contains clauses 5.1 "Namespaces and document references" and 5.2 "Name conversion". It is mapped to the following folder structure:

- + 05_mapping_xml_schemas
- + 05_top_level
- + 0501_namespaces_and_document_references
- + 0502_name_conversion
- …

5.2.2 Test case grouping

A test case typically checks a single requirement specified in [1]. However, tests for multiple requirements are possible, especially in cases when the requirements are interconnected and testing them individually would not be feasible.

Test cases consist of several files that are wrapped into folders in the lowest scope of the ATS hierarchy. The test case folders are created in the location that corresponds to their position in [1]. A folder containing test cases cannot contain folders for subclauses; top_level folders are created for testing requirements of clauses that include numbered subclauses.

Test case folder name is derived from the clause folder name in the following way:

1) The clause name is prefixed with "Pos_" for test cases that shall compile successfully and execute without runtime errors. These test cases are called positive test cases.

   Positive semantic tests shall be correct with respect to the TTCN-3 BNF, the static semantics of TTCN-3, should include correct XML schema inputs, and meet the respective text clauses of ETSI ES 201 873-9 [1]. They shall produce an ETS. If an ETS is produced and if it contains a control-part or a test case, it should be executed.

2) The clause name is prefixed with "Neg_" for test cases that either shall produce compilation errors or whose execution shall lead to a runtime error. These test cases are called negative test cases.

   Negative tests shall be correct with respect to the TTCN-3 BNF and the static semantics of TTCN-3, and should include correct XML schema inputs, but violate the semantics one specific text clause of ETSI ES 201 873-9 [1]. They may produce an ETS. If an ETS is produced and if it contains a control-part or a test case, it should be executed.

3) The clause name shall be suffixed with the low line character followed by a three-digit ordinal number of the test case. The ordinal number of the first test case defined inside a (sub)clause folder is 001 and for each following test case it increases by 1 (e.g. the ordinal number of the 4th test case is 004).

4) Positive and negative test cases are numbered separately, so there can be positive and negative test cases with the same ordinal number.

Table 1 presents test sample test case structure for clause 6.1 "Mapping of facets" of the clause 6 "Basic types".

The test cases shall conform to the following correctness rules:

- The test case identifiers and their group index do not imply the correct execution order of a TTCN-3 tool conformance test.
- Grouping and subgrouping in the ATS is realized with the help of the ATS directory structure.

Table 1: Example ATS structure of positive tests

Group	Subgroup	Group Index
Mapping of facets *Clause 6.1*	Length *Clause 6.1.1*	Pos_060101_length
	Mininclusive *Clause 6.1.7*	Pos_060107_mininclusive
	Maxinclusive *Clause 6.1.8*	Pos_060108_maxinclusive
	Maxexclusive *Clause 6.1.10*	Pos_060110_maxexclusive

Table 2: Example ATS structure of negative tests

Group	Subgroup	Group Index
Mapping of facets *Clause 6.1*	Mininclusive *Clause 6.1.1*	Neg_060101_length_001
	Maxexclusive *Clause 6.1.10*	Neg_060110_maxexclusive_001

Test case folders contain several files with the testing code. Two of these files are mandatory:

- XSD file: specifies the schema with declarations that are the subject of testing.

- TTCN-3 file: contains test description, expected result declaration and TTCN-3 module used for validation of the imported XSD declarations.

Depending on a test case type, additional files can be present as well:

- XML file: used for validation of messages generated from XSD declarations; it contains the expected encoding result and it is mandatory for positive test cases.
- Additional XSD files: used in case of various import scenarios and namespace tests.
- `test_data` folder: contains extra files for reference to validate test output.

TTCN-3 and XML file names and names of the mandatory XSD file consist of the name of the enclosing test case folder and a mandatory extension:

- <TC_folder_name>.ttcn (for TTCN-3 files)
- <TC_folder_name>.xml (for XML files)
- <TC_folder_name>.xsd (for XSD files)

In case of additional XSD files, the file name is derived from the name of the main XSD file, with low line and an additional single-digit ordinal number inserted between the test case name and the file extension:

- <TC_folder_name>_1.xml (for extra XML file)
- <TC_folder_name>_1.xsd (for extra XSD file)

EXAMPLE: Folder structure for a test case Pos_050103_imports_001:

- Containing folder: 05_mapping_xml_schemas\0501_namespaces_and_document_references\050103_imports\
  - Folder: Pos_050103_imports_001; contains files
    - Pos_050103_imports_001.ttcn
    - Pos_050103_imports_001.xsd
    - Pos_050103_imports_001_1.xsd
    - Pos_050103_imports_001.xml

5.2.3 Test case identifiers

The test case names are built up according to the following schema:

<"TC">"_"<Group index>"_"<TC number>"

where:

a) double quotes (") are used to enclose literal strings;

b) <Group index> containing positive and negative syntactic and semantic test, refers to ETSI ES 201 873-1 [10] clause numbers and names;

c) <TC number> is a running 3-digit decimal number, starting in each subgroup path with "001".

EXAMPLE: TC_Pos_060101_length_001

i) The example refers to a positive test case for string with length restriction.

ii) It is the first test case of this group/subgroup.

NOTE: The TP identifier of TC_Pos_060101_length_001 is TP_Pos_060101_length_001.

5.2.4 Naming convention inside XSD files

The XSD files shall define a target namespace unless it contains a dedicated test for rules connected with handling no target namespace in which case the target namespace shall be omitted. The target namespace shall consist of the string "schema:" followed by the schema file name without extension:

<"schema">"_"<XSD_file_name>

where:

a) double quotes (") are used to enclose literal strings;

b) XSD_file_name is the name of the XSD file name without extension.

EXAMPLE: XSD file **Pos_050103_imports_001.xsd** defines the target namespace "**schema:Pos_050103_imports_001**"

The tests that check module name transformation are the only exception to this rule. These tests might define a different namespace name according to the test case purpose.

In the XSD file, the default unprefixed namespace shall be the schema namespace http://www.w3.org/2001/XMLSchema. The target namespace shall be prefixed; preferable prefix is "ns". Only test cases testing encoding of namespace prefixes are allowed to use different namespace prefixing strategy.

5.3 ATS specification framework

5.3.1 Use of TTCN-3

5.3.1.1 General

TTCN-3, as defined in ETSI ES 201 873-1 [10], is used as the ATS specification language.

A number of requirements have been identified for the development and production of the TTCN-3 specification for the ATS:

1) Top-down design.

2) A uniquely defined testing architecture and test method.

3) Uniform TTCN-3 style and naming conventions.

4) Human-readability.

5) The TTCN-3 specification shall be feasible, implementable, compilable, and maintainable.

6) Test cases shall be designed in a way to be easily adaptable, upwards compatible with the evolution of the base protocol and protocol interworking of future releases.

7) The test declarations, data structures, and data values shall be largely reusable.

8) Modularity and modular working method.

9) Minimizing the requirements of intelligence on the emulators of the lower testers.

10) Giving enough design freedom to the test equipment manufacturers.

Fulfilling these requirements should ensure the investment of the TTCN-3 implementation vendors and users of the ATS having stable testing means for a relatively long period.

5.3.1.2 TTCN-3 naming conventions

Like in other software projects using a programming language, the use of naming conventions supports or increases:

a) the readability;

b) the detection of semantic errors;

c) the shared work of several developers;

d) the maintainability.

The naming conventions applied to Reference Test suite ATS are based on the following underlying principles:

- when constructing meaningful identifiers, the general guidelines specified for naming in Clause 9 of ETSI TS 102 351 [3] should be followed;
- the names of TTCN-3 objects being associated with standardized data types (e.g. in the base protocols) should reflect the names of these data types as close as possible (of course not conflicting with syntactical requirements or other conventions being explicitly stated);
- the subfield names of TTCN-3 objects being associated with standardized data type should also be similar to corresponding element names in the base standards (be recognizable in the local context);
- in most other cases, identifiers should be prefixed with a short alphabetic string (specified in Table 4) indicating the type of TTCN-3 element it represents;
- prefixes should be separated from the body of the identifier with an underscore ("_");
- only test case names, module names, data type names, and module parameters should begin with an upper-case letter. All other names (i.e. the part of the identifier following the prefix) should begin with a lower-case letter.

Table 4 specifies the naming guidelines for each element of the TTCN-3 language indicating the recommended prefix and capitalization.

Table 4: TTCN-3 naming convention

Language element	Naming convention	Prefix	Example	Notes
Module	Use upper-case initial letter	*none*	IPv6Templates	
TSS grouping	Use all upper-case letters as specified in Clause 7.1.2.1.1	*none*	TP_RT_PS_TR	
Item group within a module	Use lower-case initial letter	*none*	messageGroup	
Data type	Use upper-case initial letter	*none*	SetupContents	
Message template	Use lower-case initial letter	m_	m_setupInit m_setupBasic	Note 1
Message template with wildcard or matching expression	Use lower-case initial letters	mw_	mw_anyUserReply	Note 2
Signature template	Use lower-case initial letter	s_	s_callSignature	
Port instance	Use lower-case initial letter	*none*	signallingPort	
Test component ref	Use lower-case initial letter	*none*	userTerminal	
Constant	Use lower-case initial letter	c_	c_maxRetransmission	
External constant	Use lower-case initial letter	cx_	cx_macId	
Function	Use lower-case initial letter	f_	f_authentication()	
External function	Use lower-case initial letter	fx_	fx_calculateLength()	
Altstep (incl. Default)	Use lower-case initial letter	a_	a_receiveSetup()	
Test case	Use numbering as specified in Clause 5.2.3	TC_	TC_COR_0009_47_ND	
Variable (local)	Use lower-case initial letter	v_	v_macId	
Variable (defined within a component)	Use lower-case initial letters	vc_	vc_systemName	
Timer (local)	Use lower-case initial letter	t_	t_wait	
Timer (defined within a component)	Use lower-case initial letters	tc_	tc_authMin	
Module parameter	Use all upper case letters	*none*	PX_MAC_ID	Note 3
Parameterization	Use lower-case initial letter	p_	p_macId	
Enumerated Value	Use lower-case initial letter	e_	e_syncOk	

NOTE 1: This prefix is used for all template definitions which do *not* assign or refer to templates with wildcards or matching expressions, e.g. templates specifying a constant value, parameterized templates without matching expressions, etc.
NOTE 2: This prefix is used in identifiers for templates which either assign a wildcard or matching expression (e.g. ?, *, value list, ifpresent, pattern, etc.) or reference another template which assigns a wildcard or matching expression.
NOTE 3: In this case it is acceptable to use underscore as a word delimiter.

5.3.1.3 TTCN-3 comment tags

Any TTCN-3 definition in the Test Suite Repository or Library should contain embedded comment tags, according to ETSI ES 201 873-10 [2]. These comment tags can be used by tools to extract information from the TTCN-3 code to create, for example, a HTML-based reference documentation.

Comment tags which cover one or more lines should be specified using block comments, as illustrated:

```
/* ----------------------
 * @desc This line of text is now identified as a description
 *       which covers multiple lines
 * ----------------------*/
```

Comments tags specified within a single line may be specified using line comments, as illustrated:

```
// @author John Doe
```

or:

```
/* @author John Doe */
```

Table 5 lists the tags that can be used in ETSI TTCN-3 test specifications with a short description of the intended use of each tag.

NOTE: Tools may also extract other information from the TTCN-3 code based, for example, on TTCN-3 keywords. The definition of that extraction is beyond the scope of the present document.

Table 5: TTCN-3 comment tags

Tag	Description
@author	This tag should be used to specify the names of the authors or an authoring organization which either has created or is maintaining a particular piece of TTCN-3 code.
@desc	This is probably the most important of all the tags. It should be used to describe the purpose of a particular piece of TTCN-3 code. The description should be concise yet informative and describe the function and use of the construct.
@remark	This tag may be used to add additional information, such as highlighting a particular feature or aspect not covered in the description.
@img	This tag may be used to associate images with a particular piece of TTCN-3 code.
@see	This tag may be used to refer to other TTCN-3 definitions in the same or another module.
@url	This tag should be used to associate references to external files or web pages with a particular piece of TTCN-3 code, e.g. a protocol specification or standard.
@return	This tag should only be used with functions. It is used to provide additional information on the value returned by the given function.
@param	This tag is used to document the parameters of parameterized TTCN-3 definitions.
@version	This tag is used to state the version of a particular piece of TTCN-3 code.
@verdict	This tag is used to state when a TTCN-3 module passes a conformance test.
@purpose	This tag is used to state the purpose of a particular piece of TTCN-3 code.

The following provides some basic guidelines on the usage of tags for specific TTCN-3 definitions:

- Each TTCN-3 module should use the *@author*, *@version* and *@desc* tags.
- The *@desc* tag should be used with all TTCN-3 definitions. However, this should not be taken to the extreme. For example, it is probably not useful to tag literally every single constant or template declaration. It is left to the discretion of the writer to find the right level of use. At least all major constructs such as test cases and functions should have a comprehensive description:
  - when a TTCN-3 definition uses module parameters, it is also recommended to mention this explicitly in the description;
  - descriptions for behavioural constructs should mention if they set the test component verdict and also all known limitations of the construct;
  - descriptions for type definitions, e.g. component types, should mention if the type has been designed to be type compatible to another type or vice versa to be used as a basis for other type definitions.
- The @see tag should be used to make dependencies between TTCN-3 definitions which are described by a @desc tag more explicit in the documentation, e.g. if some TTCN-3 definition uses a module parameter then its TTCN-3 definition should be referenced to using a @see tag.
- Where applicable, parameterized constructions such as functions, altsteps and templates should use the *@param* and *@return* tags. The @param tags should first list the parameter name and then a brief description of how this parameter is used by the construct.
- The *@url* tag should be used to refer to the specification from which the TTCN-3 definition was derived from, e.g. a type definition could refer to a particular RFC IETF page. In some cases it may be necessary to use the *@desc* tag instead for this purpose as documents often are hard to access internally, i.e. it may only be possible to specify a reference to a complete document but impossible to point to a very specific clause in the present document.
- The *@url* and *@img* tag may be used to link to relevant documentation such as Test Purposes or original requirements or even drawings of test configurations. Generally, the corresponding Test Purpose (in the TSS&TP) and to the corresponding Requirement (in the Requirements Catalogue) should be linked from the relevant TTCN-3 test case definition.
- The *@remark* tag may be used with any TTCN-3 definition. It should be used sparingly, e.g. possibly to indicate how a TTCN-3 definition should not be used.

- The *@verdict* tag is of special importance for the present document and the ATS of annex A. Each module contains a @verdict tag (on module level) that describes when a TTCN-3 module or a set of TTCN-3 modules that comprise a TTCN-3 tool conformance test pass the conformance test. For that purpose, information about the expected tool output is encoded into the verdict tag. The overall format for the @verdict document tag is as follows:

  - @verdict pass accept/reject [expectedoutput]

  The first parameter of the @verdict document tag describes that the following information describes the criteria for a "pass" conformance verdict. The second parameter shall either be "accept" or "reject". "Accept" implies that an ETS shall be produced which may be executed. "Reject" implies that either no ETS is produced or that the TTCN-3 modules are rejected during runtime. If the second parameter is "accept", the optional third comma separated "expectedoutput" parameter is mandatory. The third parameter "expectedoutput" can adopt the following values: "noexecution" implies that verdict "pass" is assigned if ETS is produced. An execution is not required. Further possible values for the third parameter are "ttcn3verdict:none", "ttcn3verdict:pass", "ttcn3verdict:inconc", "ttcn3verdict:fail", and "ttcn3verdict:error". In these cases, a TTCN-3 conformance test passes if the TTCN-3 modules as tool input produce one of the specified TTCN-3 verdicts. A special value for "expectedoutput" parameter is "manual". This value marks that this test has to be validated manually since it cannot be detected automatically. The value "manual" should be used with caution since it can easily increase the validation time of the results.

EXAMPLE 1:

- @verdict pass reject
- @verdict pass accept, ttcn3verdict:pass

Overall, the only allowed parameter combinations are the following:

- reject
- accept, noexecution
- accept, ttcn3verdict:none
- accept, ttcn3verdict:pass
- accept, ttcn3verdict:inconc
- accept, ttcn3verdict:fail
- accept, ttcn3verdict:error
- accept, manual:"Text gives user instructions to decide when the testcase should pass."

In the usual case, each TTCN-3 module contains only one test case. In these cases, the verdict determination is clear. If the TTCN-3 file contains more than one test case, the overall conformance verdict is determined according to the TTCN-3 verdict overwrite rules applied to the results of each test case. For example, given two test cases, the first test case ends with the verdict "fail" and the second one ends with the verdict "pass". Then the overall verdict is "fail".

- The *@purpose* tag should be used with test case or module definitions depending on which definition level is more suitable to describe the corresponding conformance test purpose. The required encoding in the attached ATS of annex A has a special requirement regarding its format which is as follows:

  - @purpose documentreference, description

  The "documentreference" parameter refers to a reference to the TTCN-3 standards according to the following format:

  - part:clause

  The part refers to the part number of the respective TTCN-3 standard. The clause refers to the dot-separated clause number of the respective TTCN-3 standard.

EXAMPLE 2:

- @purpose 1:5, Ensure that when the IUT loads a module containing some definitions before the module declaration then the module is rejected.

In example 2, the purpose refers to clause 5 of ETSI ES 201 873-1 [10] and is followed by a test purpose description after the comma.

Please note that the T3Doc tags @verdict and @purpose are required for each test module inside this ATS.

5.3.2 Module parameters

```
/**
 * @desc The timeout given in seconds after which the test case will be stopped.
 */
modulepar float PX_TC_EXECUTION_TIMEOUT := 5.0;
```

Each test case module defines one module parameter: `PX_TC_EXECUTION_TIMEOUT`. It specifies the timeout in seconds used to limit test execution in the control part:

```
control {
    execute(TC_Neg_05_top_level_001(), PX_TC_EXECUTION_TIMEOUT);
}
```

5.3.3 Test case structure

The TTCN-3 file consists of a header containing test case description and a TTCN-3 module for testing the selected requirement.

The header part shall be composed of comments tagged according to [2]. The following tags are required:

- *@author*: it shall contain STF 475.
- *@version*: it shall contain the test version. When modified, the version number shall increase.
- @purpose: consist of the digit 9 (representing ETSI ES 201 873-9 [1]), followed by colon and the clause where the tested requirement is specified. After that a short description shall be present, preceded with a comma.

EXAMPLE 1: A test for a requirement from the clause 5.1.1 can have the following purpose:
** @purpose 9:5.1.1, Verify that schema with target namespace is correctly translated.

- @verdict: this tag is used for automated test execution. In case of positive tests, it typically contains "pass accept, ttcn3verdict:pass" and in case of negative tests "pass reject". However, different values are allowed too. They described in detail in the test validation tool documentation.

The header may optionally be concluded with a description of requirements that are covered by the test script.

COMPLETE HEADER SAMPLE:

```
/***********************************************
 ** @author   STF 475
 ** @version  0.0.1
 ** @purpose  9:5.1.1, Verify that schema with target namespace is correctly translated into single
module
 ** @verdict  pass accept, ttcn3verdict:pass
***********************************************/
// The following requirements are tested:
// A single XML Schema may be composed of a single or several schema element information
// items, and shall be translated to one or more TTCN-3 modules, corresponding to schema
// components that have the same target namespace. For XSD schemas with the same target
// namespace (including absence of the target namespace) exactly one TTCN-3 module shall
// be generated.
```

The TTCN-3 module shall have the same name as the test case folder. The module consists of the following parts:

- XSD import declarations
- Port and component type

- Message template
- Test case
- Control part

The import declaration connects the TTCN-3 code with the XSD declarations under test. Typically, there is just one import statement, but when testing include and import features or module naming rules, there might be more import statements.

EXAMPLE 2:

```
module Pos_050101_namespaces_001 {
   import from schema_Pos_050101_namespaces_001 language "XSD" all;
```

Port and component type definitions are always the same. The port type defines a simple message port allowing sending and receiving messages of all types. The component type definition contains a single port declaration.

Message template is a simple value template based on a XSD element definition. This template is used for validation of conversion rules specified in [1] and for checking XSD codec functionality.

EXAMPLE 3:

```
   template MyType m_msg := 1;
    type universal charstring Raw;
    type universal charstring File;
    type record of File FileList;
    type port P message {
        inout all;
    }
    type component C {
        port P p;
    }
```

The test case declaration contains the most important part of the TTCN-3: execution and validation logic. The behaviour inside test cases is explained in detail in the following clauses. The test case name shall be equal to the module name prefixed with "TC_".

```
  testcase TC_Pos_050101_namespaces_001() runs on C system C {
    var Raw v_rcv;
    var universal charstring v_matchError;
    map(self:p, system:p);

    // encode the message
    p.send(m_msg);

    alt {
        // compare the encoded message with the reference XML file
      []    p.check(receive(Raw:?) -> value v_rcv) {
        log("XML message ", v_rcv);
        if (matchFile(v_rcv, "Pos_050101_namespaces_001.xml", { "Pos_050101_namespaces_001.xsd" },
v_matchError)) {
          alt {
            // match decoded value to pass test
            [] p.receive(m_msg) {
                setverdict(pass, "Decoded value matches encoded template and reference XML");
            }
            [] p.receive {
              setverdict(fail, "XML decoding failure");
            }
          }
        } else {
          setverdict(fail, v_matchError);
        }
      }
      [] p.receive {
        setverdict(fail, "Raw decoding failure");
      }
    }
```

The control part is very similar in all test cases. It executes the only defined test case with a predefined execution time limit to prevent deadlocks.

5.3.4 External functions

5.3.4.1 External functions declared in TTCN-3

The test cases uses an external function `matchFile` to validate produced XML against the corresponding XML schema (or multiple schemas in the case of multy-XSD test case), and compare the produced XML against a reference XML file bundled with the test case.

The function is declared as:

```
external function matchFile(Raw p_textToMatch, File p_referenceXmlFile, FileList p_xsdFileList,
         out universal charstring p_matchError, File p_referenceTTCN3File := __FILE__)
    return boolean;
```

The function validates XML document contained in the charstring `p_textToMatch` and compares against the contents of the specified reference XML file and returns true if it is valid and represent the same XML structure.

Parameters of the external function are:

- `p_textToMatch` text to be compared with the UTF-8 contents of the reference XML file;
- `p_referenceXmlFile` the name of the reference XML file;
- `p_xsdFileList` the list of names of XSD files;
- `p_matchError` out parameter that contains textual description of errors in case it did not match;
- `p_referenceTTCN3File` full path of the file with the TTCN-3 test module. This path is used to find the reference XML file relative to this path, by keeping the TTCN-3 code file system independent.

Returns `true` if `p_textToMatch` and the contents of `p_referenceXmlFile` represent the same XML structure.

Two XML files are considered equal if:

1) Tree of element nodes are equal, and qualified names of elements are the same for corresponding nodes.

NOTE: for unordered subtrees corresponding to `choice` XML Schema declarations the number of child element nodes match, and there exists an ordering of nodes in the sample XML tree that matched the ordering in the reference XML tree.

2) Set of attributes for each element node are equal. Qualified names of attributes match.

3) Literal values of attributes match.

   a) The attributes of `float` type are compared by their numerical values.

4) Textual contents of elements match, whitespaces are ignored. CDATA is considered equal to plain text with equal contents.

   a) Elements of `float` type are compared by their numerical values.

5) Comments and processing instructions are ignored.

Comparison algorithm is based on XmlUnit toolkit, written in Java™. The reference implementation of the external function is implemented in Java™.

5.3.4.2 XmlDiff utility

XmlDiff utility is a helper class that implements XML comparison facility for the external function. It is an adapter for XmlUnit toolkit; its main role is to process arguments passed to the external function, prepare inputs for the reference XML file and XML string to be matched, invoke XmlDiff to perform comparison and process the comparison result.

The main entry point to XmlDiff utility is **XmlDiff** class. This class implements an algorithm to compare two XML files. It is used as a utility in the ETSI STF on conformance testing for XSD processing (ETSI ES 201 873-9 [1]).

- **Field Summary**

Fields	
Modifier and Type	**Field and Description**
`static` **`LogUtil`**	**`logger`**

- **Constructor Summary**

Constructors
Constructor and Description
`XmlDiff``(java.io.File file, java.lang.String[] xsdFileNames, java.lang.String[] xsdSearchPath)` Initialize the diff engine.
`XmlDiff``(java.lang.String referenceXmlFile, java.lang.String[] xsdFileNames, java.lang.String[] xsdSearchPath)` Initialize the diff engine.

- **Method Summary**

Methods	
Modifier and Type	**Method and Description**
`boolean`	**`diff`**`(java.lang.String input, java.lang.StringBuilder diffDetails)` Compare an XML document against the reference one.

- **Field Detail**

logger

```
public static LogUtil logger
```

- **Constructor Detail**

XmlDiff

```
public XmlDiff(java.lang.String referenceXmlFile,
       java.lang.String[] xsdFileNames,
       java.lang.String[] xsdSearchPath)
```

Initialize the diff engine. If `xsdFileNames` is `null` then the value of `xsdSearchPath` is ignored.

Parameters:

`referenceXmlFile` - path to the reference XML file

`xsdFileNames` - optional list of XSD files relevant for the reference XML file. May be `null`.

`xsdSearchPath` - optional list of folder names and/or URIs where to look for XSD files. May be `null`.

XmlDiff

```
public XmlDiff(java.io.File file,
       java.lang.String[] xsdFileNames,
       java.lang.String[] xsdSearchPath)
```

Initialize the diff engine. If `xsdFileNames` is `null` then the value of `xsdSearchPath` is ignored.

Parameters:

`file` - path to the reference XML file

`xsdFileNames` - optional list of XSD files relevant for the reference XML file. May be `null`.

`xsdSearchPath` - optional list of folder names and/or URIs where to look for XSD files. May be `null`.

- **Method Detail**

diff

```
public boolean diff(java.lang.String input,
           java.lang.StringBuilder diffDetails)
throws XmlDiffError
```

Compare an XML document against the reference one.

Parameters:

`input` - the text of the XML document

`errorMessage` - container to store details of differences. May be `null`

Returns:

`true` if the documents match and `false otherwise.`

`Throws:`

`XmlDiffError` - if an error occurs before or during diffing

5.4 ATS archive

Annex A contains the ATS archive (ts_103255v010101p0.zip file expanding to text files with TTCN-3 code, XML Schema inputs and reference XML files).

6 PIXIT conformance

A test realizer, producing an executable test suite for the Abstract Test Suite (ATS) specification, is required, as specified in ISO/IEC 9646-4 [5], to produce an augmented partial PIXIT proforma conformant with this partial PIXIT proforma specification.

An augmented partial PIXIT proforma which conforms to this partial PIXIT proforma specification shall, as a minimum, have contents which are technically equivalent to annex B. The augmented partial PIXIT proforma may contain additional questions that need to be answered in order to prepare the Means Of Testing (MOT) for a particular Implementation Under Test (IUT).

A test laboratory, offering testing for the ATS specification contained in annex A, is required, as specified in ISO/IEC 9646-5 [6], to further augment the augmented partial PIXIT proforma to produce a PIXIT proforma conformant with this partial PIXIT proforma specification.

A PIXIT proforma which conforms to this partial PIXIT proforma specification shall, as a minimum, have contents which are technically equivalent to annex B. The PIXIT proforma may contain additional questions that need to be answered in order to prepare the test laboratory for a particular IUT.

7 ATS conformance

The test realizer, producing a Means Of Testing (MOT) and Executable Test Suite (ETS) for the present document, shall comply with the requirements of ISO/IEC 9646-4 [5]. In particular, these concern the realization of an Executable Test Suite (ETS) based on each ATS. The test realizer shall provide a statement of conformance of the MOT to the present document.

An ETS which conforms to the present document shall contain test groups and test cases which are technically equivalent to those contained in the ATS in annex A. All sequences of test events comprising an abstract test case shall be capable of being realized in the executable test case. Any further checking which the test system might be capable of performing is outside the scope of the present document and shall not contribute to the verdict assignment for each test case.

Test laboratories running conformance test services using this ATS shall comply with ISO/IEC 9646-5 [6].

A test laboratory which claims to conform to this ATS specification shall use an MOT which conforms to this ATS.

Annex A (normative): Abstract Test Suite (ATS)

A.1 The ATS in TTCN-3 core (text) format

The TTCN-3 modules have been produced using the Testing and Test Control Notation (TTCN-3) according to ETSI ES 201 873-9 [1].

The TTCN-3 core (text) representation corresponding to this ATS is contained in several ASCII files contained in archive ts_103255v010101p0.zip which accompanies the present document.

Annex B (normative): Partial IXIT proforma

B.1 Introduction

This partial IXIT proforma contained in the present document is provided for completion, when the related Abstract Test Suite is to be used against the Implementation under Test (IUT).

The completed partial IXIT will normally be used in conjunction with the completed ICS, as it adds precision to the information provided by the ICS.

As all features of the TTCN-3 core standard specification to ETSI ES 201 873-9 [1] are mandatory, there is no test suite parameterization and hence there are also no IXIT tables.

Each test suite test contains one IXIT named PX_TC_EXECUTION_TIMEOUT used for configuring the timeout after which a test case will be stopped when started with the control part.

History

Document history		
V1.1.1	March 2015	Publication