Final draft ETSI ES 202 786 V1.4.1 (2017-03)

Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Support of interfaces with continuous signals

Reference

RES/MTS-202786 ed141ContSign

Keywords

interface, testing, TTCN-3

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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

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 (https://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 final draft ETSI Standard (ES) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS), and is now submitted for the ETSI standards Membership Approval Procedure.

The use of underline (additional text) and strike through (deleted text) highlights the differences between base document and extended documents.

The present document relates to the multi-part standard ETSI ES 201 873 covering the Testing and Test Control Notation version 3, as identified below:

Part 1: "TTCN-3 Core Language";

Part 4: "TTCN-3 Operational Semantics";

Part 5: "TTCN-3 Runtime Interface (TRI)";

Part 6: "TTCN-3 Control Interface (TCI)";

Part 7: "Using ASN.1 with TTCN-3";

Part 8: "The IDL to TTCN-3 Mapping";

Part 9: "Using XML schema with TTCN-3";

Part 10: "TTCN-3 Documentation Comment Specification".

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 defines the "Continuous Signal support" package of TTCN-3. TTCN-3 can be used for the specification of all types of reactive system tests over a variety of communication ports. Typical areas of application are protocol testing (including mobile and Internet protocols), service testing (including supplementary services), module testing, testing of APIs, etc. TTCN-3 is not restricted to conformance testing and can be used for many other kinds of testing including interoperability, robustness, regression, system and integration testing. The specification of test suites for physical layer protocols is outside the scope of the present document.

TTCN-3 packages are intended to define additional TTCN-3 concepts, which are not mandatory as concepts in the TTCN-3 core language, but which are optional as part of a package which is suited for dedicated applications and/or usages of TTCN-3.

This package defines concepts for testing systems using continuous signals as opposed to discrete messages and the characterization of the progression of such signals by use of **streams**. For both the production as well as the evaluation of continuous signals the concept of **mode** is introduced. Also, the signals can be processed as **history**-traces. Finally, basic mathematical functions that are useful for analyzing such traces are defined for TTCN-3. It is thus especially useful for testing systems which communicate with the physical world via sensors and actuators.

While the design of TTCN-3 package has taken into account the consistency of a combined usage of the core language with a number of packages, the concrete usages of and guidelines for this package in combination with other packages is outside the scope of the present document.

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 referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://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 201 873-1 (V4.9.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

[2] ETSI ES 201 873-4 (V4.6.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics".

[3] ETSI ES 201 873-5 (V4.8.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".

[4] ETSI ES 201 873-6 (V4.9.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

[5] ISO/IEC 9646-1: "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework; Part 1: General concepts".

[6] ETSI ES 202 785 (V1.3.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Behaviour Types".

[7] ETSI ES 202 781 (V1.3.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Configuration and Deployment Support".

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

[i.2] 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.3] 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".

[i.4] ETSI ES 201 873-10: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 10: TTCN-3 Documentation Comment Specification".

[i.5] 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.6] ETSI ES 202 782: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: TTCN-3 Performance and Real Time Testing".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI ES 201 873-1 [1], ETSI ES 201 873-4 [2], ETSI ES 201 873-5 [3], ETSI ES 201 873-6 [4] and ISO/IEC 9646-1 [5] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI ES 201 873-1 [1], ETSI ES 201 873-4 [2], ETSI ES 201 873-5 [3], ETSI ES 201 873-6 [4] and ISO/IEC 9646-1 [5] apply.

4 Package conformance and compatibility

The package presented in the present document is identified by the package tag:

- `"TTCN-3:2012 Support for Testing Continuous Signals"` - to be used with modules complying with the present document.

For an implementation claiming to conform to this package version, all features specified in the present document shall be implemented consistently with the requirements given in the present document and in ETSI ES 201 873-1 [1], ETSI ES 201 873-4 [2], ETSI ES 201 873-5 [3] and ETSI ES 201 873-6 [4].

The package presented in the present document is compatible to:

ETSI ES 201 873-1 (V4.9.1) [1]

ETSI ES 201 873-4 (V4.6.1) [2]

ETSI ES 201 873-5 (V4.8.1) [3]

ETSI ES 201 873-6 (V4.9.1) [4]

ETSI ES 202 785 (V1.3.1) [6]

ETSI ES 202 781 (V1.3.1) [7]

ETSI ES 201 873-7 [i.1]

ETSI ES 201 873-8 [i.2]

ETSI ES 201 873-9 [i.3]

ETSI ES 201 873-10 [i.4]

ETSI ES 202 784 [i.5]

ETSI ES 202 782 [i.6]

If later versions of those parts are available and should be used instead, the compatibility to the package presented in the present document has to be checked individually.

5 Package concepts for the core language

5.0 General

Systems can communicate its data or signals, either in discrete form (e.g. as an integer value) or in continuous form (e.g. real values). With respect to this difference signals are classified into four categories. The categories distinguish whether the time and value domain of a signal is of discrete or continuous nature:

1) Analogue signals are continuous in the time and value domain. Analogue signals are the most 'natural' signal category, characterized by physical units (e.g. current, voltage, velocity) and measured with sensors. Typical examples of the physical quantities used in the area of embedded system development are the vehicle velocity, the field intensity of a radio station etc. Analogue signals can be described as a piecewise function over time (e.g. vx = f (t)).

2) Time quantified signals are discrete signals in the time domain. The signal values are defined only at predetermined time points (sampling points). Typical examples of time quantified signals are the time-value pairs of a recorded signal. A typical representation of a time quantified signal is a series or an array of real numbers. Even if the original signal is a synthetic function it can only be reconstructed from a time quantified signal with considerable mathematical effort.

3) Value quantified signals are time-continuous signals with discrete values. Typical examples of a value quantified signal are data that are derived from analogue signals and which are dedicated to further processing, e.g. an A/D converted sensor signal that is provided to an electrical control unit.

4) Digital signals are discrete on the time and value domain. If the set of possible signal values includes only two elements, one speaks about binary signals. Typical examples of binary signals are switching positions or flags.

Thus on a theoretical level, continuous and discrete evolution of time and values have to be distinguised. In a discrete system, the changes of states are processed at fixed and finite time steps. In a continuous system state changes occur for infinitesimally small time steps. Important mathematical models for continuous systems are ordinary differential equations. A mixed system, which shows continuous and discrete dynamics, is known as a hybrid system. Hybrid systems can be modelled with hybrid automatons. Examples for systems that show such variable dynamics are often found in the area of embedded control systems e.g. in the automotive and aircraft industry.

In the general case, a test description notation for embedded software systems shall support all of four categories of signals mentioned above. TTCN-3 currently supports the signal categories (2) and (4). The extension of the language with respect to a support of the signal categories (1) and (3) is the content of the present document.

TTCN-3 is a procedural testing language, thus test behaviour is defined by algorithms that typically send messages to ports and receive messages from ports. For the evaluation of different alternatives of expected messages, or timeout events, the port queues and the timeout queues are frozen when the evaluation starts. This kind of snapshot semantics guarantees a consistent view on the test system input during an individual evaluation step. Whereas the snapshot semantics provides means for a pseudo parallel evaluation of messages from several ports, there is no notion of simultaneous stimulation and time triggered evaluation. To enhance the core language to the requirements of continuous and hybrid behaviour the following are introduced:

- the notions of time and sampling;
- the notions of streams, stream ports and stream variables;
- the definition of an automaton alike control flow structure to support the specification of hybrid behaviour.

5.1 Time and Sampling

5.1.0 General

The TTCN-3 extensions defined in this package adopt the concept of a global clock and enhance it with the notion of sampling and sampled time. As in TTCN-3, all time values are denoted as float values and represent time in seconds. For sampling, simple equidistant sampling models as well as dynamic sampling models are supported.

On technical level an equidistant sampling model of the form **`t=k*bdelta,`** where **`t`** describes the time progress, **`d`** specifies the number of executed sampling steps and, **`bdelta`** yields the minimal achievable step size for a given test system, is used as an overall basis to model equidistant samplings with larger step size or dynamic sampling.

The basic sampling with its minimal step size **`bdelta`** is a property of a concrete test system and not intended to be specified as part of the test case specification. However, as a consequence of this underlying model, a test system is able to execute user defined samplings if and only if all specified sampling rates at test specification level provide step sizes that are multiples of **`bdelta`**.

When using the TTCN-3 extension defined in this package, each reference to time, either used for the definition and evaluation of signals but as well by means of ordinary TTCN-3 timers, is considered to be completely synchronized to the global clock and the base sampling.

5.1.1 The now operator

For the specification of time-dependent signal sequences, it is necessary to be able to track the passage of time. The access of time is guaranteed by a globally available clock whose current value can be accessed by means of the **now** operator. Time progress starts at the beginning of each test case execution, thus time values are related to the start of the test case execution.

Syntactical Structure

now

Semantic Description

Evaluation of the **now** operator yields the current value of the clock which is the duration of time since the start of the currently running test case.

Restrictions

The **now** operator shall only be applied from within a test case, i.e. by test cases, functions and altsteps executed on test components. The **now** operator shall neither directly nor indirectly be called by TTCN-3 control part.

Example

EXAMPLE:

```
 // Use of now to retrieve the actual time since the test case has started
var float actualTime := now;
```

5.1.2 Define the default step size for sampling

For sampling, a globally valid base sampling rate defined by the test system is provided. In addition, sampling rates can be set separately and as part of the test specification by means of **`stepsize`** attribute.

Syntactic Structure

`stepsize` ***`StepSizeValue`***

Semantic Description

The ***`StepSizeValue`*** is a string-literal which shall contain a decimal number. This number interpreted as seconds is used as the default rate of sampling values over the stream ports to which are affected by this **`stepsize`** attribute. The actual sampling rate of a specific port can be changed dynamically with the **`delta`** operation.

Restrictions

A **`stepsize`** attribute can only appear in a with-annotation. A **`stepsize`** attribute can be applied to individual modules, test cases, groups, component types and stream port types and effects either the statements that are contained in one of these entities or in case of component types and stream port types the respective instances.

Examples

EXAMPLE 1:

```
// sets the stepsize for a module
module myModule{
…
} with {stepsize " 0.0001" };
```

EXAMPLE 2:

```
// sets the stepsize for a testcase
testcase myTestcase() runs on myComponent{
…
} with {stepsize " 0.0001" };
```

EXAMPLE 3:

```
// sets the stepsize for all instances of the port type StreamOut
type port StreamOut stream { out float} with {stepsize " 0.0001" };
```


5.2 Data streams

5.2.0 General

In computer science the term data stream is used to describe a continuous or discrete sequence of data. Normally the length of a stream cannot be established in advance. The data rate, i.e. the number of samples per time unit, can vary. Data streams are continuously processed and are particularly suited to represent dynamically evolving variables over a course of time. Thus, streams are an ideal representation of the different discrete and continuous signals mentioned in the beginning of clause 5.

While in standard TTCN-3 interactions between the test components and the SUT are realized by sending and receiving messages through ports, the interaction between continuous systems can be represented by means of so called streams. In contrast to scalar values, a stream represents the whole allocation history applied to a port. In computer science, streams are widely used to describe finite or infinite data flows. To represent the relation to time, so called timed streams are used. Timed streams additionally provide timing information for each stream value and thus enable the traceability of timed behaviour. The TTCN-3 extension defined by this package provides timed streams. In the following, the term measurement (record) to denote the unity of a stream value and the related timing in timed streams will be used. Thus, concerning the recording of continuous data, a measurement record represents an individual measurement, consisting of a stream value that represents the data side and timing information that represents the temporal perspective of such a measurement.

In this TTCN-3 extension, two different but not complementary representations of timed data streams are introduced. The term timed considers the fact that the time and value domain of a signal are of interest. As a consequence, a stream is considered to consist of a sequence of samples. Each sample provides information about the timing and the value perspective of the sample.

of the sample.

1) Static perspective: The static perspective provides a direct mapping between a timed stream and the TTCN-3 data structures **`record`** and **`record of`**. This kind of mapping is referred to below as the static representation of a data stream and allows random access to all elements of the data stream.

2) Dynamic perspective: To provide dynamic online access to data streams, the existing concepts of TTCN-3 port type and port are extended to provide access to data streams and their content. A so called **`stream port`** references exactly one data stream and provides access to the dynamically changing values of the referenced data stream.

NOTE: To represent streams in the present document, a tabular notation is used. The table has two rows by which the first one represents the value perspective of a stream and the second represents the temporal perspective. The temporal perspective is defined by means of timestamps that are synchronized with the overall clock. The columns represent the samples of the stream.

EXAMPLE:

Value	1.2	1.4	1.5	1.7	1.7	1.5	1.2	1.0	1.1	1.4	1.5	1.2	1.0	1.1	1.4
Timestamp	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4

The example shows a stream with the length of 1.4 seconds and float values that change between 1.0 and 1.5.

5.2.1 Data Streams: static perspective

A TTCN-3 data stream can be mapped directly to existing TTCN-3 data structures. The mapping considers each stream to be represented by means of a TTCN-3 **record of** data structure. This structure itself consists of individual entities, so called *samples*, each sample representing either a measurement on an incoming stream or stimulus that is dedicated to be applied to an outgoing stream.

A sample itself is represented by means of a TTCN-3 **record** data structure. The record consists of two fields. It has two fields of type **`float`**. The first field with the name `value` represents what is called the value of a stream. Its data type should be aligned with the data type of the corresponding stream. The second field denotes the temporal perspective of a sample. It denotes the temporal distance to the preceding sample (the sampling step size `delta`). The second field is of type **`float`** and represents time values that have the physical unit second. Example 1 shows the exemplary definition of a data structure to specify individual samples.

EXAMPLE 1:

```
type record Sample{
      float value,
      float delta
}
```

Given such a structure, a timed data stream of an arbitrary data type is modelled as a record of samples.

EXAMPLE 2:

```
type record of Sample MyStreamType;
```

The static representation of data streams can be used for the online and offline evaluation of streams as well as for the piecewise in-memory definition of streams or stream templates, which are to be applied to stream ports in the subsequent test case execution. Thus, the static representation of streams can be used to assess incoming streams and to define outgoing or reference streams and template streams mostly by means of ordinary TTCN-3 operations and control structures and as such provide an ideal interface between ordinary TTCN-3 concepts and the concepts defined in this package. The following example shows a short specification of a sampled stream.

EXAMPLE 3:

```
var MyStreamType myStreamVar := {
       {value:=0.0, delta:=0.1},
       {value:=0.2, delta:=0.2},
       {value:=0.1, delta:=0.1},
       {value:=0.0, delta:=0.3}
}
```

If the stream definition from above is applied to an outgoing stream port directly with the beginning of a test case, the result will look as follows.

EXAMPLE 4:

Value	0.0	0.0	0.2	0.1	0.0
Timestamp	0	0.1	0.3	0.4	0.7

Each stream port is initialized with a value that defines the valuation of a stream at time 0.0. Thus the first sample in Example 4 is not defined by the specification in Example 3 but by the base initialization of the stream port.

NOTE 1: In order to create larger streams a manual specification approach is not feasible. In this case, the data processing capabilities of TTCN-3 can be used. This allows to programmatically/algorithmically construct the dedicated record structures.

NOTE 2: The data structures presented in this section serve for illustration purposes only. They show how timed data streams can be mapped to standard TTCN-3 data structures and thus can be processed easily by using the existing TTCN-3 language features and operators. The TTCN-3 extensions provided in this package do not include type declarations from above.

5.2.2 Data Streams: dynamic perspective

5.2.2.0 General

In standard TTCN-3 ports are used for the communication among test components and between test components and the SUT. To be able to initiate, modify and evaluate a stream based communication between the entities of a test system, this package extends the concepts of standard TTCN-3 port types and ports with the notion of *stream-based communication* and *stream ports.* Stream ports are the endpoints of a stream based communication. Thus stream ports in TTCN-3 embedded are used to provide access to streams, their values and the respective timing information. A stream port references exactly one data stream and thus provides access to the respective stream values and timing information.

5.2.2.1 Defining stream port types

The TTCN-3 port concept of message-based and procedure-based ports is extended with stream-based ports. Stream ports support stream-based communication.

Syntactical Structure

```
type port PortTypeIdentifier stream "{"
 { ( ( in | out | inout ) StreamValueType [";"] )
 ( map param "(" { FormalValuePar [","] }+ ")"  [";"] ) |
  ( unmap param "(" { FormalValuePar [","] }+ ")"  [";"] ) }
"}"
```

Semantic Description

Stream port types shall be declared by using the keyword **stream**. Stream ports are directional. The directions are specified by the keywords **in** (for the in direction), **out** (for the out direction) and **inout** (for both directions).

The specified *StreamValueType* references the type of values which can be sent or received (depending on the direction of the port) over ports of the type *PortTypeIdentifier*.

Like message and procedure ports, stream ports can use map and unmap parameters to pass additional information to the system adapter.

Restrictions

Each stream port type definition shall have one and only one entry indicating the allowed type together with the allowed communication direction:

a) Stream port type definition shall always contain exactly one stream value definition.

b) At most one map parameter list should be defined for a port type.

c) At most one unmap parameter list should be defined for a port type.

Example

EXAMPLE:

```
  // Stream-based port which allows stream values of type float to be received
type port StreamIn stream { in float }

  //Stream-based port which allows stream values of type float to be sent
type port StreamOut stream { out float }

  //Stream-based port with map and unmap parameter definitions
type port StreamOut stream
{
      in float;
      map param (integer p_par1, integer p_par2);
      unmap param (integer p_par1);
}
```

5.2.2.2 Declaration and instantiation of stream ports

The declaration of stream-based ports is similar to the declaration of message-based and procedure-based ports. The **component** type declares which ports are associated with a component. A component type can have ports with different communication characteristics (e.g. stream-based ports, message-based ports, and procedure based). All ports are instantiated together with the component that owns the port, i.e. when the component is created.

Outgoing stream ports start to emit stream values directly after the component, which contains the respective stream port, has been started. The same applies for incoming stream ports. They start receiving data directly after their component has been started. Both incoming and outgoing stream ports are updated for each sampling step. If no explicit step size is defined by means of step size annotations on module level, test case level, port type level, etc. the port is initially sampled with the test systems' base sampling, which is the smallest available step size.

Outgoing stream ports may already be initialized before its first use, so that their values before the start of their component are defined. The initialization occurs in the context of their declaration.

Outgoing stream ports, when they are not explicitly initialized, are automatically initialized with implicit default values. The implicit default values for the various TTCN-3 basic data types can be found in table 1.

Table 1

float	integer	boolean	charstring	bitstring	octetstring
0.0	0	FALSE	""	'0'B	'00'O

The initial stream port value for outgoing stream port applies to the time point 0.0 and for the following sample steps as long as no other stream value is set. The value initialization for incoming streams is in responsibility of the data provider. Hence either the system adapter or the emitting component (in case of a PTC) is responsible to initialize the streams.

Syntactical Structure

```
  port StreamPortTypeReference
{ StreamPortIdentifier [ ":=" StreamDefaultValue ]  [","] }+ [";"]
```