ETSI ES 201 873-11 V4.7.1 (2017-06)

Methods for Testing and Specification (MTS);
The Testing and Test Control Notation version 3;
Part 11: Using JSON with TTCN-3

Reference

DES/MTS-00201873-11ed471JSON

Keywords

JSON, language, 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

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

The present document is part 11 of a multi-part deliverable. Full details of the entire series can be found in part 1 [1].

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 rules to define schemas for JSON data structures in TTCN-3, to enable testing of JSON-based systems, interfaces and protocols, and the conversion rules between TTCN-3 [1] and JSON [2] to enable exchanging TTCN-3 data in JSON format between different systems.

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: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

[2] IETF® RFC 7159: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at http://www.rfc-editor.org/rfc/rfc7159.txt.

[3] ISO/IEC 10646:2014: "Information technology -- Universal Coded Character Set (UCS)".

NOTE: Available at http://standards.iso.org/ittf/PubliclyAvailableStandards/c056921_ISO_IEC_10646_2012.zip.

[4] IEEE™ 754: "IEEE Standard for Floating-Point Arithmetic".

[5] 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".

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] IETF® draft-zyp-json-schema-04: "JSON Schema: core definitions and terminology".

NOTE: Available at http://tools.ietf.org/html/draft-zyp-json-schema-04.

[i.2] IETF® draft-fge-json-schema-validation-00: "JSON Schema: interactive and non interactive validation".

NOTE: Available at http://tools.ietf.org/html/draft-zyp-json-schema-04.

[i.3] World Wide Web Consortium W3C® Recommendation: "XML Schema Part 1: Structures".

NOTE: Available at http://www.w3.org/TR/xmlschema11-1.

[i.4] World Wide Web Consortium W3C® Recommendation: "XML Schema Part 2: Datatypes".

NOTE: Available at http://www.w3.org/TR/xmlschema11-2.

[i.5] 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.6] 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.7] 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.8] 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".

[i.9] 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.10] ETSI ES 202 785: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Behaviour Types".

[i.11] ETSI ES 202 786: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Support of interfaces with continuous signals".

[i.12] ETSI ES 202 789: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; TTCN-3 Language Extensions: Extended TRI".

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] apply.

3.2 Abbreviations

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

ASN.1	Abstract Syntax Notation no. 1
IO	ASN.1 Information Object
JSON	JavaScript Object Notation
SUT	System Under Test
TTCN-3	Testing and Test Control Notation version 3
UCS	Universal Coded Character Set
USI	UCS Sequence Identifier
UTF-8	Unicode Transformation Format-8
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Introduction

An increasing number of distributed applications use the JSON to exchange data for various purposes like data bases queries or updates or event telecommunications operations such as provisioning. The JSON specification [2] defines the syntax and encoding for JSON types and defined literals, but no semantics is defined. JSON does not have a schema specification, like the XML Schema Definition Language used for XML documents (see [i.3] and [i.4]).

NOTE: Though an IETF draft proposal exists for JSON schema (see [i.1] and [i.2]), it has not reached the RFC status.

The core language of TTCN-3 is defined in ETSI ES 201 873-1 [1] and provides a full text-based syntax, static semantics and operational semantics. Other parts of the ETSI ES 201 873 series are defining its use with other specification languages like ASN.1 [5], IDL [i.5], or XSD [i.6] as shown in figure 1, while other documents as ETSI ES 202 781 [i.7], ETSI ES 202 782 [i.8], ETSI ES 202 784 [i.9], ETSI ES 202 785 [i.10], ETSI ES 202 786 [i.11] and ETSI ES 202 789 [i.12] specify language extensions and thus can define additional rules to the JSON/TTCN-3 mapping defined in the present document.

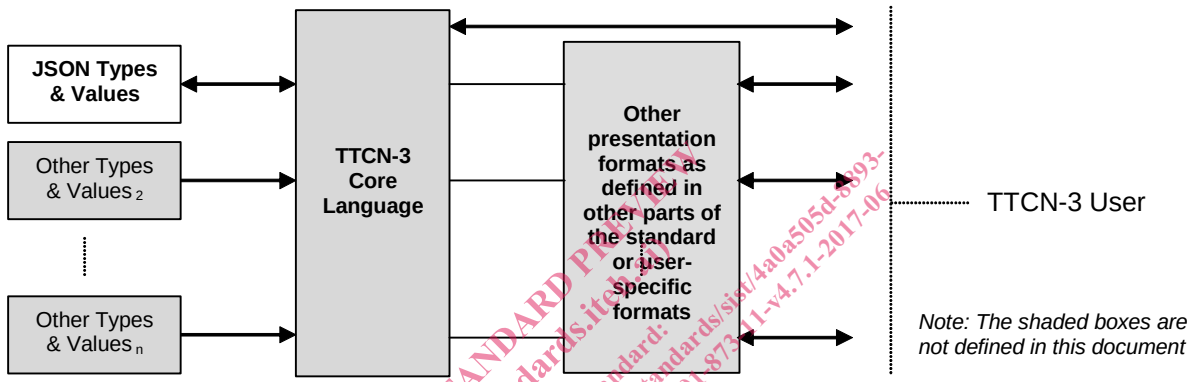

Figure 1: User's view of the core language and the various presentation formats

In the context of TTCN-3, JSON can be used for different purposes:

1) TTCN-3 can be used as a JSON Schema definition language that allows generating JSON values from TTCN-3 and consuming and evaluating received JSON values, i.e. enables testing of JSON-based interfaces and protocols.

2) To exchange type and data information between the TTCN-3 test system and systems written in other languages like Java, C, C++, Python, etc. In this way TTCN-3 test systems can be used as a subsystem of a more complex test system; for example, the TTCN-3 system receiving contents of messages to be sent to an SUT, encode and send a message, receive and process the response and report the result to the other system.

Consequently, there is a need to specify mappings between JSON and TTCN-3 for the above purposes.

5 Conformance and compatibility

For an implementation claiming to support the use of TTCN-3 as a JSON schema language, all features specified in clause 6 of the present document shall be implemented consistently with the requirements given in clause 6 and Annex B of the present document and in ETSI ES 201 873-1 [1].

For an implementation claiming to support the exchange of TTCN-3-based data between systems, and not supporting using ASN.1 with TTCN-3, all features specified in clause 7 of the present document, with the exception of the mapping of the objid type in clause 7.2.11 shall be implemented consistently with the requirements given in clause 7 and Annex B of the present document and in ETSI ES 201 873-1 [1]. Implementations claiming the support of using ASN.1 with TTCN-3, shall in addition support features in clause 7.2.11 and clause 8 of the present document.

The language mappings presented in the present document is compatible to:

- ETSI ES 201 873-1 [1], version 4.9.1.

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

6 Using TTCN-3 as JSON Schema

6.1 Approach

JSON [2] defines a limited set of JSON types and literal values. The clauses below define the TTCN-3 types that can be used to specify a Schema for any JSON interface specification. The TTCN-3 types defined in the clauses below will allow to use the same set of values as JSON permits. Annex A provides a TTCN-3 module containing all TTCN-3 definitions specified in these clauses. The **`JSON`** module in Annex A shall either explicitly be present in TTCN-3 test suites or TTCN-3 tools shall support these types implicitly. This is left as a tool implementation option.

JSON in many cases allows different encoding options for the same value. These may be controlled by the JSON encoding instructions specified in Annex B. JSON encoding instructions may be added to TTCN-3 types and fields by using TTCN-3 variant attributes (see ETSI ES 201 873-1 [1], clause 27.5).

6.2 Validation of JSON Values

For further study.

6.3 Name conversion rules

The current version of the JSON specification [2] uses names to identify JSON object members only. The present document defines JSON to TTCN-3 name conversion rules that shall be used when using TTCN-3 to specify a schema for a JSON interface (see for example clause 6.4.4). When the JSON and the TTCN-3 names differ after applying the rules in this clause, the "name as …" encoding instruction shall be used to identify the exact JSON name. To ensure compatibility with future versions and automatic conversions, the rules specified in this clause should always be applied.

JSON names can be identical to TTCN-3 reserved words, can contain characters not allowed in TTCN-3 identifiers or allowed to be identical, when the corresponding TTCN-3 names are required to be unique, in which case the JSON names shall be processed by the rules below to obtain the corresponding TTCN-3 identifiers.

The following character substitutions shall be applied, in order that each character string being mapped to a TTCN-3 name, where each substitution (except the first) shall be applied to the result of the previous transformation:

a) any character except "A" to "Z" (Latin Capital Letter A to Latin Capital Letter Z), "a" to "z" (Latin Small Letter A to Latin Small Letter Z), "0" to "9" (Digit Zero to Digit Nine), and "_" (Low Line) shall be removed;

b) a sequence of two or more "_" (Low Line) characters shall be replaced with a single "_" (Low Line);

c) "_" (Low Line) characters occurring at the beginning or at the end of the name shall be removed;

d) if a character string starts with a digit (Digit Zero to Digit Nine), it shall be prefixed with an "**x**" (Latin Small Letter X) character;

e) if a character string is empty, it shall be replaced by "**x**" (Latin Small Letter X);

f) if the TTCN-3 name being generated is identical to a previously generated TTCN-3 identifier in the same scope, then a postfix shall be appended to the character string generated by the above rules. If a field name of a TTCN-3 structured type is clashing with a type's name used in the same structured type, the field's name shall be postfixed. The postfix shall consist of a "_" (Low Line) followed by an integer. This integer shall be the least positive integer such that the new identifier is different from the identifier of any previously generated identifier in the same scope (i.e. the first postfix applied by this mechanism is "_1"). TTCN-3 names that are one of the TTCN-3 keywords (see clause A.1.5 of ETSI ES 201 873-1 [1]) or names of predefined functions (see clause 16.1.2 of ETSI ES 201 873-1 [1]) after applying the postfix to clashing names, shall be suffixed by a single "_" (Low Line) character.

6.4 Mapping of JSON Values

6.4.1 JSON Numbers

JSON numbers are represented as base 10 decimal digits containing a mandatory integer component that can be prefixed with an optional minus sign, and can be followed by a fraction part, an exponent part or both. Leading zeros are not allowed. JSON does not distinguish numbers based on their value sets like integers and reals, like other languages do. No special values (as –infinity, infinity or NaN) are allowed.

In the general case, JSON numbers shall be mapped by using the following TTCN-3 type:

```
type float Number (!-infinity .. !infinity) with {
  variant "JSON:number"
  }
```

When the JSON interface specification requires a number to conform to the ANSI/IEEE 754 [4] floating-point number specification, the IEEE 754 floats useful types of clause E.2.1.4 of ETSI ES 201 873-1 [1] can be used in the context of JSON encoding, in which case, by default, the given useful type will constrain the value set and the encoding of the JSON value according to this clause. The JSON encoding instructions in this case can be applied to fields of IEEE 754 useful types.

By default, i.e. without any encoding instruction applied, the form of the JSON representation of *JSON.Number* is a tool implementation option (i.e. the number of fraction digits, using the exponent part, etc.)

To make defining JSON Schemas in TTCN-3 easier, the present document, in addition to the generic mapping of JSON numbers, also specifies a TTCN-3 type that may be used where the interface specification allows only numbers without the fraction and the exponent parts:

```
type integer Integer (-infinity .. infinity) with {
    variant "JSON:integer"
}
```

Attempts to decode a JSON number value with either a fraction or an exponent part or both into this *JSON.Integer* type shall cause a decoding failure.

In addition to the generic encoding instructions like "normalize" and "name as …", the following specific instructions shall be applicable to types and fields of *JSON.Number* and the IEEE 754 useful types:

- fractionDigits see clause B.3.5
- useMinus see clause B.3.6

NOTE: The beginning character of the exponent part can be both "e" and "E". This is not controlled by any of the encoding instructions but left as a tool implementation option.

and to types and fields of *JSON.Integer* types:

- useMinus see clause B.3.6

6.4.2 JSON Strings

A JSON string is a sequence of zero or more Unicode characters, enclosed in a pair of quotation mark characters ("""", **`char(U22)`**). Any characters may be escaped by the escape sequence: "\u<HHHH>", where <HHHH> represents four hexadecimal digits, but the icharacters: quotation mark (""", **`char(U22)`**), reverse solidus ("\", **`char(U5C)`**) and all C0 control characters (**`char(U0)`** through **`char(U1F)`**) shall be escaped.

Alternatively, the short, two-character escape sequences defined in table 1 can be used to escape some of the characters.

Table 1: Short character escape sequences

Character's name	Character code	Short escape sequence
quotation mark	**`char(U22)`**	\"
reverse solidus	**`char(U5C)`**	\\
solidus	**`char(U2F)`**	\/
backspace	**`char(U8)`**	\b
form feed	**`char(UC)`**	\f
line feed	**`char(UA)`**	\n
carriage return	**`char(UD)`**	\r
horizontal tab	**`char(U9)`**	\t

By default, it is a tool implementation option which form of escaping is used, which may be overridden by the "escape as …" encoding instruction.

NOTE 1: Note that the JSON module in Annex A defines useful TTCN-3 constants for the characters listed above.

The following TTCN-3 type shall be used to map JSON strings to TTCN-3:

```
type universal charstring String with {
  variant "JSON:string"
}
```

NOTE 2: Though Unicode and ISO/IEC 10646 [3] do not necessarily contain the same set of characters at all points in time, JSON strings are expressed using the TTCN-3 universal charstring type.

In addition to the generic encoding instructions like "normalize" and "name as …", the following specific encoding instructions are applicable to **`JSON.String`** types:

- escape as … see clause B.3.7

EXAMPLE: String encoding examples

If:

```
const JSON.String c_string1 := <actual value> with {variant "escape as short"};
```

then

<actual value>	JSON character sequence	UTF-8 serialization of the JSON value	Note
"abcd"	"abcd"	226162636422	
"ab\cd"	"ab\\cd"	2261625C5C636422	
"ab/cd"	"ab\/d"	2261625C2F636422	
"ab" & char(U7) & "cd"	"ab\u0007cd"	2261625C75303030 37636422	
"ab" & char(U7) & cu_ht & "cd"	"ab\u0007\u0009cd"	2261625C75303030 375C753030303963 6422	The horizontal tab character is encoded according to the encoding instruction of the referenced definition ("escape as usi") and not according to the referencing definition ("escape as short")