ETSI ES 203 022 V1.2.1 (2018-05)

Methods for Testing and Specification (MTS);

The Testing and Test Control Notation version 3;

TTCN-3 extension: Advanced Matching

**ETSI Standard**

Reference

RES/MTS-203022ed121

Keywords

conformance, 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***

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.  
The content of the PDF version shall not be modified without the written authorization of ETSI.  
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2018.

All rights reserved.

**DECT**TM, **PLUGTESTS**TM, **UMTS**TM and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.  
**3GPP**TM and **LTE**TM are trademarks of ETSI registered for the benefit of its Members and  
of the 3GPP Organizational Partners.  
**oneM2M** logo is protected for the benefit of its Members.  
**GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

# 5.5 Encodable Values

Encodable values are all TTCN-3 entities that can be encoded. They are comprised of all values, value templates and encodable value templates. Encodable value templates are special templates which can be encoded but not be used as a value in any other way. Thus, the only operations allowed on encodable value templates are operations that involve encoding, i.e. the sending operations and the encvalue operations.

***Restrictions***

1. Encodable value templates shall not be assigned to variables or parameters of unrestricted templates or of templates with the present, value or omit restrictions.
2. Encodable value templates shall not be used as operands for expressions.
3. Encodable value templates shall not be used as value parameters.
4. Encodable value templates shall not be used in receive operations or in operations which require the value restriction.

## 5.5.1 The **encvalue** Template Restriction

***Syntactical Structure***

**template** "(" **encvalue** ")" **| template** "(" **encvalue** "**,**" **omit** ")"

***Semantic Description***

To allow to differentiate between normal **value** restricted templates and encodable value templates, the template restriction (**encvalue)** shall be used. It is possible to define templates with that restriction as well as use the **template(encvalue)** modifier for the declaration of variables, formal parameters and return clauses. Additionally, to describe encodable templates that can be used in place of optional fields, the **(encvalue,omit)** restriction shall be used.

Templates with the (encvalue) restriction can be derived from templates with value or encvalue restriction by usage of modifies. Template with the (encvalue,omit) restriction can be derived from templates with value, encvalue or omit restriction by usage of modifies.

Any template that contains at least one part with an encvalue restriction also fulfills the encvalue restriction.

The following relationships between different kinds of templates exist:

* template(value) is subset of template(encvalue)
* template(encvalue) is subset of template(encvalue, omit)
* template(omit) is subset of template(encvalue, omit)
* template(encvalue) is not subset of unrestricted template
* template(encvalue, omit) is not subset of unrestricted template

Any template expression that is part of a subset of another template kind can be used also as the more general kind. Using a template with the **encvalue** restriction as a template without that restriction shall produce an error.

***Examples***

EXAMPLE 1:

template(encvalue) charstring msg\_foobar := “foobar”;

function f\_encvalue(template(encvalue) charstring p\_msg := msg\_foobar)

return template(encvalue) charstring {

var template(encvalue, omit) charstring v\_msg := p\_msg; // allowed

return p\_msg; // allowed

}

EXAMPLE 2:

template(omit) charstring msg\_omit := “foobar”;

template(encvalue, omit) charstring msg\_encvalue\_omit := msg\_omit; // allowed

function f\_encvalue(template(encvalue, omit) charstring p\_msg := msg\_encvalue\_omit) // allowed

return template(encvalue) charstring {

var template(encvalue) charstring v\_msg := p\_msg; // not allowed

var template charstring v\_msg2 := p\_msg; // not allowed

return p\_msg; // not allowed

}

## 5.5.2. Encoding Mutation Annotation

In encodable value templates, it is allowed to add a mutation annotation to parts of the template which is applied during encoding of the annotated part as a post-processing step to the original result produced by the encoder for that part.

***Syntactical Structure***

[ *TemplateInstance* ] (**@mutation** | **@mutation\_o** | **@mutation\_unichar** [ "(" *StringEncoding* ")" ]) *Expression*

***Semantic Description***

The family of mutation annotations **@mutation**, **@mutation\_o** and **@mutation\_unichar** are template expressions which conform to the **encvalue** template restriction.

If the *TemplateInstance* has the **omit** template restriction, then the resulting encodable value template has the restriction **(encvalue,omit)**. If the *TemplateInstance* has the **(value)** or **(encvalue)** restriction, the resulting encodable value template has the restriction **(encvalue)**. Encodable templates with the **(encvalue)** restriction can be safely assigned to mandatory fields while templates with the **(encvalue,omit)** restriction can also be assigned to optional fields.

If one of the mutation annotation keywords occurs to the right of a *TemplateInstance*, then the *Expression* on the right side of the mutation annotation keyword can use the keyword **value** as an implicit formal parameter to reference the encoded value of that *TemplateInstance*. If the *Expression* does not need to reference the encoded value, then the *TemplateInstance* may be omitted.

If the **@mutation** keyword is used, then the **value** keyword refers to an expression of type **bitstring** and the *Expression* shall evaluate to a value of type **bitstring**.

If the **@mutation\_o** keyword is used, then the **value** keyword refers to an expression of type **octetstring** and the *Expression* shall evaluate to a value of type **octetstring**.

If the @**mutation\_unichar** keyword is used, then the **value** keyword refers to an expression of type **universal charstring** and the *Expression* shall evaluate to a value of type **universal charstring**. If a different string encoding than the default “UTF-8” is used for the universal charstring, then this string encoding is given as an additional *StringEncoding* operand in parenthesis after the **@mutation\_unichar** keyword.

When an encoder processes a value template that is a mutation annotation with a *TemplateInstance*, it will first encode that *TemplateInstance* into a sub-message. It will then transform that sub-message into a TTCN-3 string value of the appropriate type (depending on which mutation annotation is used) and then invoke the evaluation of the mutation *Expression*, using the transformed string value as an actual parameter of the formal parameter **value**. The result of the evaluation is transformed back to a sub-message which is then used instead of the original sub-message as part of the resulting message.

When an encoder processes a value template that is a mutation annotation without a *TemplateInstance*, it will evaluate the mutation *Expression* and transform the resulting value to a sub-message which is then used as the part of the message corresponding to the encoded value.

If the **@mutation\_o** keyword is used, the sub-message is transformed into a left-aligned **octetstring** before transformation, so that if the sub-message does not have a bit-length divisible by 8, the appropriate amount of padding bits are the least significant bits of the least significant octet of the **octetstring**. The bit-content of the whole octetstring that is the result of the evaluation will be used as the resulting sub-message.

If the **@mutation\_unichar** keyword is used, the sub-message is transformed depending on the given *StringEncoding* into a **univeral charstring**. The transformed sub-message must be byte-aligned and have a bit-length that is consistent with the given *StringEncoding* and otherwise an error will be produced. The result of the evaluation is a **universal charstring** that is transformed into a sub-message by using the given *StringEncoding* to encode it into a byte-aligned binary representation.

***Restrictions***

1. The *Expression* shall conform to the restrictions given in clause 16.4.1 and shall not use any functions with a runs on clause.
2. The *TemplateInstance* shall be an encodable value.
3. The **value** keyword shall not be used inside the *Expression* if no *TemplateInstance* is given.

The following sections shall be changed in the following ways:

### 22.2.1 The Send operation

***Restrictions***

a) The *TemplateInstance* shall be an encodable value, i.e. the use of matching mechanisms such as *AnyValue* is not allowed.

### 22.3.1 The Call operation

***Restrictions***

b) All **in** and **inout** parameters of the signature shall be an encodable value, i.e. the use of matching mechanisms such as *AnyValue* is not allowed.

### 22.3.3 The Reply operation

***Restrictions***

1. <unchanged>
2. <unchanged>
3. All **out** and **inout** parameters of the signature shall be encodable values i.e. the use of matching mechanisms such as *AnyValue* is not allowed.
4. <unchanged>
5. <unchanged>
6. <unchanged>
7. The *TemplateBody* in the **value** clause shall be an encodable value and it shall be type‑compatible with the return type specified in the signature of the procedure to which the **reply** operation belongs.
8. <unchanged>
9. <unchanged>

### 22.3.5 The Raise operation

The following sentences:

The value part of the **raise** operation consists of the signature reference followed by the exception value.

Exceptions are specified as types. Therefore the exception value may either be derived from a template conforming to the template(value) restriction or be the value resulting from an expression (which of course can be an explicit value). The optional type field in the value specification to the **raise** operation shall be used in cases where it is necessary to avoid any ambiguity of the type of the value being sent.

… are to be replaced with the following text:

The value part of the **raise** operation consists of the signature reference followed by the exception *TemplateInstance*.

Exception types are specified in the referenced *Signature* declaration. The exception given to the raise operation shall be an encodable value. In cases where it is necessary to avoid any ambiguity of the type of the value being sent, the *TemplateInstance* shall use the inline template syntax with the exception type as prefix.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5 and shown in table 16, the following restrictions apply:

a) <unchanged>

b) <unchanged>

c) <unchanged>

d) <unchanged>

e) <unchanged>

f) The *TemplateInstance* shall be an encodable value.

g) <unchanged>

## C.5.1 The encoding function

**encvalue**(**in template (encvalue)** any\_typeinpar,

**in** **universal charstring** encoding\_info := "",

**in** **universal** **charstring** dynamic\_encoding := "") **return** **bitstring**

The **encvalue** function encodes an encodable value into a bitstring. The returned bitstring represents the encoded value of inpar, however, the TTCN-3 test system need not make any check on its correctness. The optional encoding\_info parameter is used for passing additional encoding information to the codec and, if it is omitted, no additional information is sent to the codec.

The optional dynamic\_encoding parameter is used for dynamic selection of **encode** attribute of the **inpar** value for this single **encvalue** call. The rules for dynamic selection of the **encode** attribute are described in clause 27.9.

In addition to the general error causes in clause 16.1.2, error causes are:

* Encoding fails due to a runtime system problem (i.e. no encoding function exists for the actual type of inpar).

## C.5.3 The encoding to universal charstring function

**encvalue\_unichar**(**in template** (**encvalue**)any\_typeinpar,

**in** **charstring** string\_serialization := "UTF-8",

**in** **universal charstring** encoding\_info := "",

**in universal charstring** dynamic\_encoding := "")

**return universal charstring**

The **encvalue\_unichar** function encodes an encodable value into a universal charstring. The returned universal charstring represents the encoded value of inpar, however, the TTCN-3 test system need not make any check on its correctness. If the optional string\_serialization parameter is omitted, the default value "UTF-8" is used. The optional encoding\_info parameter is used for passing additional encoding information to the codec and, if it is omitted, no additional information is sent to the codec.

The optional dynamic\_encoding parameter is used for dynamic selection of **encode** attribute of the **inpar** parameter for this single encvalue\_unichar call. The rules for dynamic selection of the **encode** attribute are described in clause 27.9.

The following values (see ISO/IEC 10646 [2]) are allowed as string\_serialization actual parameters (for the description of the UCS encoding scheme see clause 27.5):

1. "UTF-8"
2. "UTF-16"
3. "UTF-16LE"
4. "UTF-16BE"
5. "UTF-32"
6. "UTF-32LE"
7. "UTF-32BE"

The serialized bitstring shall not include the optional signature (see clause 10 of ISO/IEC 10646 [2], also known as byte order mark).

In case of "UTF-16" and "UTF-32" big-endian ordering shall be used (as described in clauses 10.4 and 10.7 of ISO/IEC 10646 [2]).

The specific semantics of this function are explained by the following TTCN-3 definition:

**function** encvalue\_unichar(**in** **template**(**encvalue**) any\_type inpar,

**in** **charstring** enc

**in** **universal** **charstring** encoding\_info := "",

**in universal charstring** dynamic\_encoding := "")

**return** **universal** **charstring** {

**return** **oct2unichar**(**bit2oct**(**encvalue**(inpar, encoding\_info, dynamic\_encoding)), enc);

}

The encvalue\_unichar function first invokes the encvalue function in order to encode the encodable value passed in the inpar parameter to a bitstring. The bitstring is then converted to an octetstring by the bit2oct function and subsequently to a universal charstring using the oct2unichar function. The string\_serialization parameter defines how the encoded octets (in fact the encoded bitstring received from the codec) contain the characters. The universal charstring value is then returned as the result of the encvalue\_unichar function.

In addition to the general error causes in clause 16.1.2, error causes are:

* Encoding fails due to a runtime system problem (i.e. no encoding function exists for the actual type of inpar).
* The given string encoding is not recognized.

## C.5.5 The encoding to octetstring function

**encvalue\_o**(**in template (encvalue)** any\_typeinpar,

**in** **universal charstring** encoding\_info := "",

**in** **universal** **charstring** dynamic\_encoding := "") **return octetstring**

The **encvalue\_o** function encodes an encodable value into an octetstring. The returned octetstring represents the encoded value of inpar, however, the TTCN-3 test system need not make any check on its correctness. The optional encoding\_info parameter is used for passing additional encoding information to the codec and, if it is omitted, no additional information is sent to the codec.

The optional dynamic\_encoding parameter is used for dynamic selection of **encode** attribute of the **inpar** value for this single **encvalue\_o** call. The rules for dynamic selection of the **encode** attribute are described in clause 27.9.

In addition to the general error causes in clause 16.1.2, error causes are:

* Encoding fails due to a runtime system problem (i.e. no encoding function exists for the actual type of inpar).