TD <>

Draft ETSI ES **DES/MTS-00138** V0.0.1 (2011-09)

Methods for Testing and Specification (MTS);

Testing and Test Control Notation version 3;

Extension Package: Extended TRI

<

**ETSI SPECIFICATION**

Reference

<Workitem>

Keywords

<keywords>

***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***

Individual copies of the present document can be downloaded from:  
<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the 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 <http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:  
<http://portal.etsi.org/chaircor/ETSI_support.asp>

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute yyyy.

All rights reserved.

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

# 7 TRI extensions for the package

#### 5.6.3.3 triRnd → xtriRnd

|  |  |
| --- | --- |
| **Signature** | FloatValue xtriRnd(in TriComponentIdType componentId, in FloatValue seed) |
| **In Parameters** | componentId identifier of the component for which to generate the random number |
|  | seed the seed to be used for generation of the random number or null |
| **Out Parameters** | n.a. |
| **Return Value** | The generated float random number |
| **Constraints** | This operation is called by the PA to generate a random number in the context of an external function. |
| **Effect** | A random number is generated in the scope of the component identified by the given component ID using the given seed (if any) according to the specification of the predefined rnd function defined in ES 201 873‑1 [2]. |

## 7.7 Changes to 6 Java language mapping

#### 6.5.3.2 TriPlatformTE

The triPlatformTE interface is mapped to the following Java interface:

// TriPlatform

// PA ‑> TE

package org.etsi.ttcn.tri;

public interface XTriPlatformTE {

// Ref: TRI‑Definition 5.6.2.5

public void triTimeout(TriTimerId timerId);

// Error handling

// Ref: TRI‑Definition 5.2.2

public void triPAErrorReq (String message);

// Ref: TRI-Definition 5.6.3.2

public TriCompnentId triSelf();

// Ref: TRI-Definition 5.6.3.3

public FloatValue xtriRnd(TriComponentId componentId, FloatValue seed);

}

## 7.8 Changes to 7 C language mapping

### 7.2.4 TRI operation mapping

TriStatus xtriMapParam

(const TriPortId\* compPortId,

const TriPortId\* tsiPortId,

const TciParameterList\* parameterList)

TriStatus xtriUnmapParam

(const TriPortId\* compPortId,

const TriPortId\* tsiPortId,

const TciParameterList\* parameterList)

TriStatus xtriSend

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const Value\* sutAddress,

const Value\* sendMessage)

TriStatus xtriSendBC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const Value\* sendMessage)

TriStatus xtriSendMC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const ValueList\* sutAddresses,

const Value\* sendMessage)

void xtriEnqueueMsg

(const TriPortId\* tsiPortId,

const Object\* sutAddress,

const TriComponentId\* componentId,

const Object\* receivedMessage)

TriStatus xtriCall

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const Value\* sutAddress,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList)

TriStatus xtriCallBC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList)

TriStatus xtriCallMC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const ValueList\* sutAddresses,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList)

TriStatus xtriReply

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const Value\* sutAddress,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList,

const Value\* returnValue)

TriStatus xtriReplyBC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList,

const Value\* returnValue)

TriStatus xtriReplyMC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const ValueList\* sutAddresses,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList,

const Value\* returnValue)

TriStatus xtriRaise

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const Value\* sutAddress,

const TriSignatureId\* signatureId,

const Value\* exception)

TriStatus xtriRaiseBC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const TriSignatureId\* signatureId,

const Value\* exception)

TriStatus xtriRaiseMC

(const TriComponentId\* componentId,

const TriPortId\* tsiPortId,

const ValueList\* sutAddresses,

const TriSignatureId\* signatureId,

const Value\* exception)

void xtriEnqueueCall

(const TriPortId\* tsiPortId,

const Object\* sutAddress,

const TriComponentId\* componentId,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList)

void xtriEnqueueReply

(const TriPortId\* tsiPortId,

const Object\* sutAddress,

const TriComponentId\* componentId,

const TriSignatureId\* signatureId,

const TciParameterList\* parameterList,

const Value\* returnValue)

void xtriEnqueueException

(const TriPortId\* tsiPortId,

const Object\* sutAddress,

const TriComponentId\* componentId,

const TriSignatureId\* signatureId,

const Object\* exception)

TriStatus xtriExternalFunction

(const TriFunctionId\* functionId,

TciParameterList\* parameterList,

Value\* returnValue)

Value xtriConvert

(Object\* value,

Type\* decodingHypothesis)

TFloat xtriRnd(TriComponentId \*componentId, TFloat\* seed)

## 7.9 Changes to 8 C++ language mapping

### 8.6.4 TriPlatformTE

This interface consists of operations that are necessary to implement the communication of the platform, in which the testcase is running, with the TTCN-3 ETS. It is mapped to the following pure virtual class:

class TriPlatformTe {

public:

//Destructor.

virtual ~TriPlatformTE ();

//Notify the timeout of the timer.

virtual void triTimeout (const TriTimerId \*timerId)=0;

//Called by PA in unrecoverable error situations.

virtual void triPAError (const Tstring &message)=0;

//Called by PA inside external function

virtual TriComponentId \*triSelf ()=0;

//Generate random number.

virtual FloatValue\* xtriRnd (const TriComponentId \*componentId, const FloatValue \*seed)=0;

}

## 7.10 Changes to 9 C# language mapping

#### 9.5.2.4 ITriPlatformTE

The **ITriPlatformTE** interface is defined as follows:

public interface ITriPlatformTE {  
 // Ref: TRI-Definition clause 5.6.2.5  
 void TriTimeout(ITriTimerId timerId);  
 // Ref: TRI Definition clause 5.2.2

void TriPAErrorReq(string message);

// Ref: TRI Definition clause 5.6.3.2

ITriComponentId TriSelf();

// Ref: TRI Definition clause 5.6.3.3

FloatValue XTriRnd(ITriComponentId componentId, FloatValue seed);

}