International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.166
(11/2007)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – Testing and Test Control Notation (TTCN)

Testing and Test Control Notation version 3: TTCN-3 control interface (TCI)

Recommendation ITU-T Z.166

ITU-T Z-SERIES RECOMMENDATIONS

LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	Z.100–Z.109
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
User Requirements Notation (URN)	Z.150–Z.159
Testing and Test Control Notation (TTCN)	**Z.160–Z.179**
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-machine interfaces for the management of telecommunications networks	Z.360–Z.379
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Processing environment architectures	Z.600–Z.609

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Z.166

Testing and Test Control Notation version 3:
TTCN-3 control interface (TCI)

Summary

ITU-T Recommendation Z.166 specifies the control interfaces for TTCN-3 test system implementations. The TTCN-3 Control Interfaces provide a standardized adaptation for management, test component handling and encoding/decoding of a test system to a particular test platform. This Recommendation defines the interfaces as a set of operations independent of a target language.

The interfaces are defined to be compatible with the TTCN-3 standards (see clause 2). The interface definition uses the CORBA Interface Definition Language (IDL) to specify the TCI completely. Clauses 8 and 9 present language mappings for this abstract specification to the target languages Java and ANSI C. A summary of the IDL-based interface specification is provided in Annex A.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

CONTENTS

Recommendation ITU-T Z.166

Testing and Test Control Notation version 3:
TTCN-3 control interface (TCI)

1   Scope

This Recommendation specifies the control interfaces for TTCN-3 test system implementations. The TTCN-3 Control Interfaces provide a standardized adaptation for management, test component handling and encoding/decoding of a test system to a particular test platform. This Recommendation defines the interfaces as a set of operations independent of a target language.

The interfaces are defined to be compatible with the TTCN-3 standards (see clause 2). The interface definition uses the CORBA Interface Definition Language (IDL) to specify the TCI completely. Clauses 8 and 9 present language mappings for this abstract specification to the target languages Java and ANSI C. A summary of the IDL-based interface specification is provided in Annex A.

2   References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Z.161]   ITU-T Recommendation Z.161 (2007), *Testing and Test Control Notation version 3: TTCN-3 core language*.

[ITU-T Z.164]   ITU-T Recommendation Z.164 (2007), *Testing and Test Control Notation version 3: TTCN-3 operational semantics*.

[ITU-T Z.165]   ITU-T Recommendation Z.165 (2007), *Testing and Test Control Notation version 3: TTCN-3 runtime interface (TRI)*.

[ITU-T X.290]   ITU-T Recommendation X.290 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – General concepts*.

ISO/IEC 9646-1:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts*.

[W3C Primer]   W3C Recommendation, *XML Schema Part 0: Primer*.

NOTE – See at http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[W3C structures]   W3C Recommendation, *XML Schema Part 1: Structures*.

NOTE – See at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[W3C Datatypes]   W3C Recommendation, *XML Schema Part 2: Datatypes*.

NOTE – See at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

3 Definitions and abbreviations

3.1 Definitions

This Recommendation defines the following terms:

3.1.1 abstract test suite (ATS): Test suite composed of abstract test cases.

3.1.2 codec: Encoder/decoder entity used for encoding and decoding data to be transmitted and received, respectively.

3.1.3 coding/decoding (CD): Entity that administers the value and type handling including encoding and decoding in the TTCN-3 test system.

3.1.4 component handling (CH): Entity that administers the handling of test components in the TTCN-3 test system.

3.1.5 communication port: Abstract mechanism facilitating communication between test components.

NOTE – A communication port is modelled as a FIFO queue in the receiving direction. Ports can be message-based, procedure-based or a mixture of the two.

3.1.6 control component: Component that executes the behaviour of the control part of a TTCN-3 module.

3.1.7 executable test suite (ETS): Refer to [ITU-T X.290].

3.1.8 implementation eXtra information for testing (IXIT): Refer to [ITU-T X.290].

3.1.9 platform adaptor (PA): Entity that adapts the TTCN-3 Executable to a particular execution platform.

NOTE – The platform adaptor creates a single notion of time for a TTCN-3 test system, and implements both, explicit and implicit, timers as well as external functions.

3.1.10 real test system interface: Refer to [ITU-T X.290].

3.1.11 system under test (SUT): Refer to [ITU-T X.290].

3.1.12 SUT adaptor (SA): Entity that adapts the TTCN-3 communication operations with the SUT based on an abstract test system interface. It implements the real test system interface.

3.1.13 testing and test control notation (TTCN-3): Refer to [ITU-T X.290].

3.1.14 test case: Refer to [ITU-T X.290].

3.1.15 test event: Either sent or received test data (message or procedure call) on a communication port that is part of the test system interface as well as timeout events of timers.

3.1.16 test management (TM): Entity which provides a user interface to as well as the administration of the TTCN-3 test system.

3.1.17 test logging (TL): Entity which provides logging information about test execution (including also the information provided by the TTCN-3 log statement).

3.1.18 test management and control (TMC): Set of entities providing test management and control; consists of the test management (TM), the component handling (CH), the test logging (TL) and the coding/decoding (CD).

NOTE – The TMC is an implementation of TCI.

3.1.19 test system: Refer to [ITU-T X.290].

3.1.20 test system interface (TSI): Test component that provides a mapping of the ports available in the (abstract) TTCN-3 test system to those offered by a real test system.

3.1.21 TTCN-3 executable (TE): Part of a test system that deals with interpretation or execution of a TTCN-3 ETS.

3.1.22 TTCN-3 control interfaces (TCI): Three interfaces that define the interaction of the TTCN-3 Executable with the test management, the coding and decoding, and the test component handling in a test system.

3.1.23 TTCN-3 runtime interface (TRI): Interface that defines the interaction of the TTCN-3 Executable with the SUT and platform adaptors in a test system.

3.2   Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ATS   Abstract Test Suite

CD   (External) Coding/Decoding

CH   Component Handler

ETS   Executable Test Suite

IDL   Interface Definition Language

IXIT   Implementation eXtra Information for Testing

MTC   Main Test Component

OMG   Object Management Group

PA   Platform Adaptor

PTC   Parallel Test Component

SA   SUT Adaptor

SUT   System Under Test

TCI   TTCN-3 Control Interfaces

TE   TTCN-3 Executable

TL   Test Logging

TM   Test Management

TMC   Test Management and Control

TRI   TTCN-3 Runtime Interface

TSI   Test System Interface

TTCN-3   Testing and Test Control Notation Version 3

4   Introduction

This Recommendation consists of two distinct parts, the first part describing the structure of a TTCN-3 test system implementation and the second part presenting the TTCN-3 Control Interfaces specification.

The first part introduces the decomposition of a TTCN-3 test system into four main entities:

•   Test Management and Control (TMC).

•   TTCN-3 Executable (TE).

•   SUT Adaptor (SA).

•   Platform Adaptor (PA).

The TMC consists itself of three entities: Test Management (TM), Coder/Decoder (CD), and Test Component Handler (CH). In addition, the interaction between these entities, i.e., the corresponding interfaces, is defined.

The second part of this Recommendation specifies the TTCN-3 Control Interfaces (TCI). The interfaces are defined in terms of operations implemented as part of one entity and called by other test system entities. For each operation, the interface specification defines associated data structures, the intended effect on the test system and any constraints on the usage of the operation. Note that these interface specifications only define interactions between the TE and TM, TE and CD, and TE and CH. For interactions between the TE and SA and the TE and PA, please refer to the TTCN-3 Runtime Interface specification ([ITU-T Z.165]).

5   Compliance

The minimum required for a TCI compliant TTCN-3 test system is to adhere to the interface specification stated in this Recommendation. The TTCN-3 semantics in the test system must adhere to the operational semantics defined in [ITU-T Z.164]. In addition, one language mapping must be supported. For example, if a vendor supports Java, the TCI operation calls and implementations, which are part of the TTCN-3 executable, must comply with the IDL to Java mapping specified in this Recommendation. For the logging interface, the XML mapping can be used instead of the Java or the C mapping.

6   General structure of a TTCN-3 test system

A TTCN-3 test system can be thought of conceptually as a set of interacting entities. Each entity implements specific test system functionality. These entities:

- manage test execution;
- interpret or execute compiled TTCN-3 code;
- realize proper communication with the SUT;
- administer types, values and test components;
- implement external functions; and
- handle timer operations.

6.1 Entities in a TTCN-3 test system

The structure of a TTCN-3 test system implementation is illustrated in Figure 1.
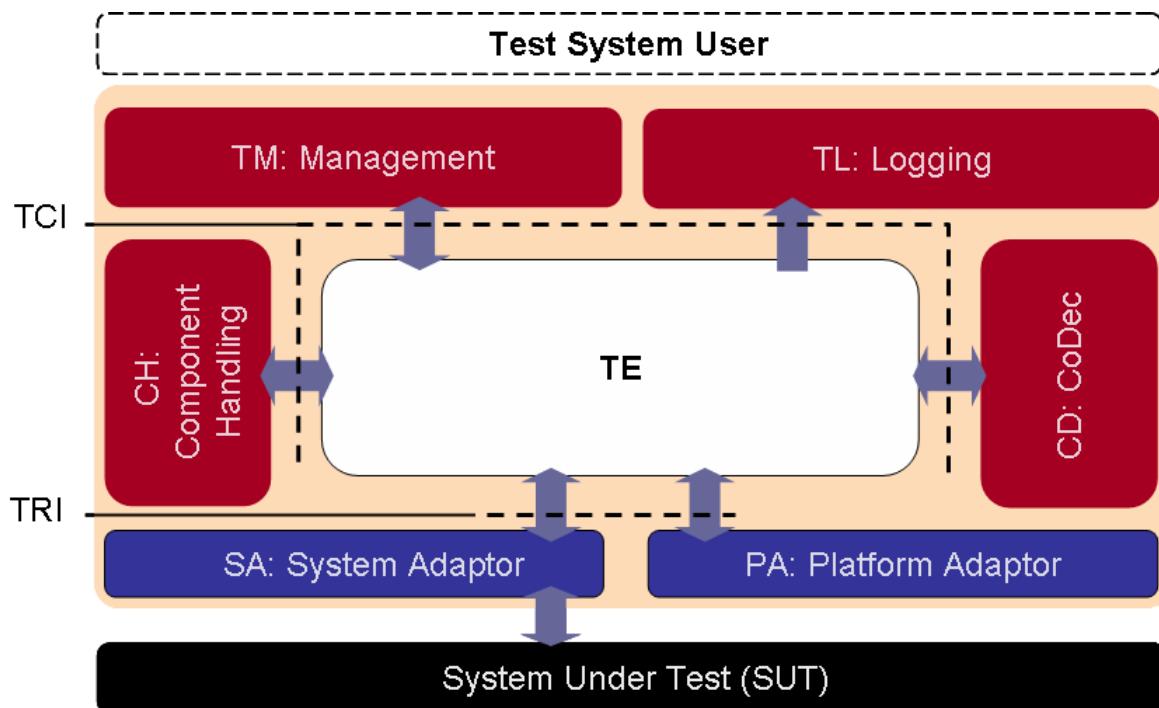
Figure 1 – General structure of a TTCN-3 test system

As shown in Figure 1, the TTCN-3 Executable (TE), also referred to as the Executable Test Suite (ETS), interprets and executes TTCN-3 modules. Various TE structural elements can be identified: control, behaviour, components, types, values and queues. The structural elements within the TE represent functionality that is defined within a TTCN-3 module or by the TTCN-3 standard ([ITU-T Z.161]) itself. For example, the structural element "Control" represents the control part within a TTCN-3 module, while the structural element "Queues" represents the requirement on a TTCN-3 Executable that each port of a test component maintains its own port queue. While the first is specified within a TTCN-3 module, the latter is required by the TTCN-3 specification.

Refinement of the TE, as shown in Figure 1, is provided as an aid in defining the TTCN-3 Control Interfaces. The TE would typically correspond in a test system implementation either to the executable code produced by a TTCN-3 compiler or by a TTCN-3 interpreter.

The TE may be executed in a centralized or in a distributed manner. That is, on a single test device or across several test devices respectively. Although the structural entities of the TE implement a complete TTCN-3 module, single structural entities might be distributed over several test devices.

The TE implements a TTCN-3 module on an abstract level. The other entities of a TTCN-3 test system make these abstract concepts concrete. For example, the abstract concept of sending a message or receiving a timeout cannot be implemented within the TE. The remaining part of the test system implements the encoding of the message and its sending over concrete physical means or measuring the time and determining when a timer has expired, respectively.

The SA and PA and their interaction with the TE are defined in [ITU-T Z.165]. The TCI specification defines the interaction between the TE and the TMC.

The logging interface provides logging capabilities to all elements of the test system architecture, i.e., the TE, the TM, the CH, the CD, the SA and the PA are able to log information on the test execution via TL. Figure 2 represents a more detailed view on TL.
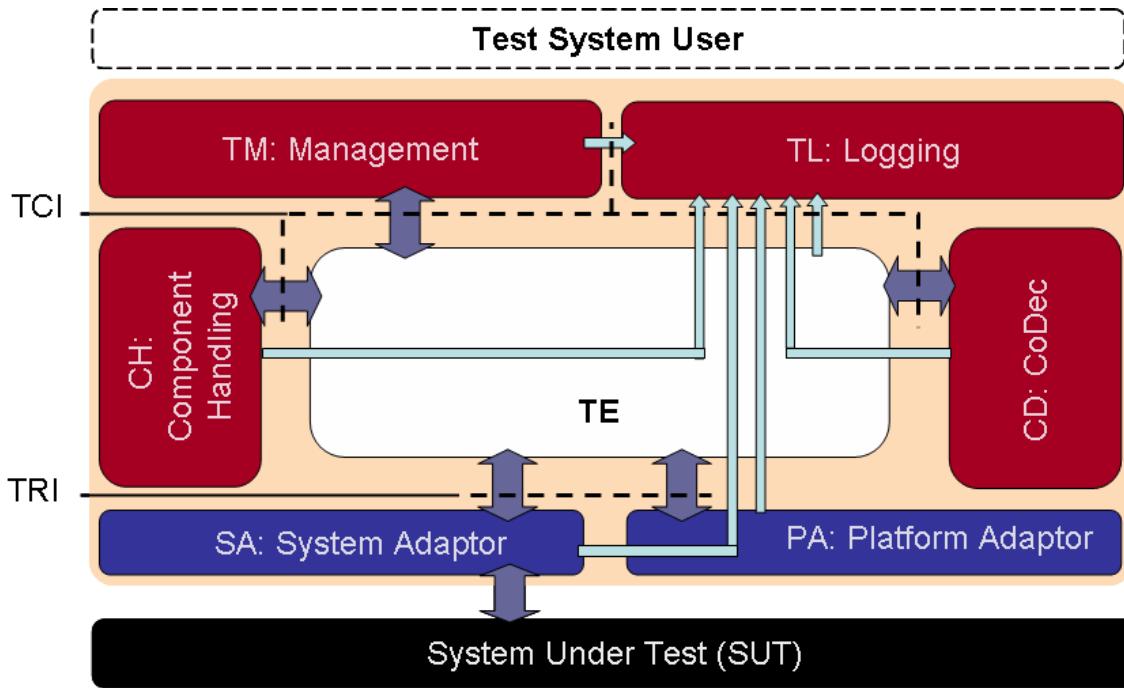
Figure 2 – Detailed view on TL

6.1.1 Test management and control (TMC)

The TMC entity includes functionality related to management of:

- test execution;
- components;
- encoding and decoding; and
- logging.

6.1.1.1 Test management (TM)

The TM entity is responsible for the overall management of a test system. After the test system has been initialized, test execution starts within the TM entity. The entity is responsible for the proper invocation of TTCN-3 modules, i.e., propagating module parameters such as IXIT information to the TE if necessary. Typically, this entity would also implement a test system user interface.

6.1.1.2 Coding and decoding (CD)

The CD entity is responsible for the external encoding and decoding of TTCN-3 values into bitstrings suitable to be sent to the system under test. Whenever external codecs are used, the TE determines which codecs shall be used. It passes the TTCN-3 data to the appropriate encoder to obtain the encoded data. Received data is decoded in the CD entity by using the appropriate decoder, which translates the received data into TTCN-3 values.

6.1.1.3 Component handling (CH)

The TE can be distributed among several test devices. The CH implements communication between distributed test system entities. The CH entity provides the means to synchronize test system entities which might be distributed onto several nodes.

NOTE 1 – Nodes and test devices are used as synonyms.

The general structure of a test system distributed among several nodes is depicted in Figure 3.
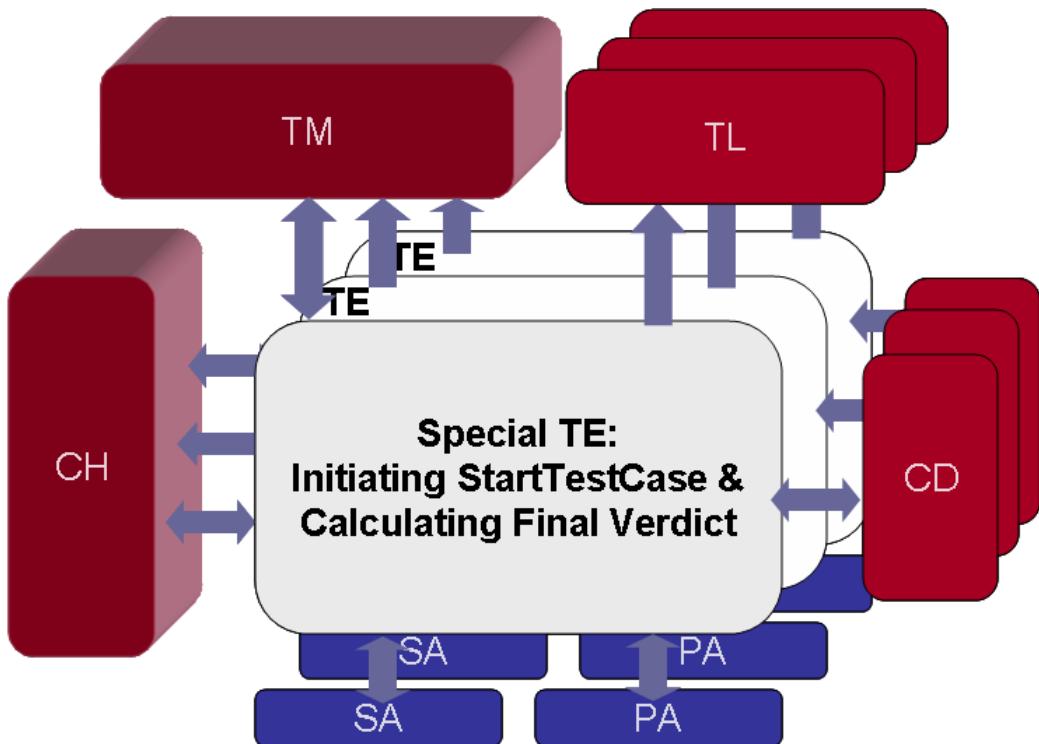
Figure 3 – General structure of a distributed TTCN-3 test system

Each node within a test system includes the TE, SA, PA, CD and TL entities. The entities CH and TM mediate the test management and test component handling between the TEs on each node. The TE which starts a test case is a special TE. It shall calculate the final test case verdict. Besides this, all TEs are handled the same.

NOTE 2 – As stated in [ITU-T Z.164], a test system executes at most one test case at a given point in time, i.e., a TTCN-3 module cannot execute multiple test cases at the same time.

The creation of the MTC, PTCs and the control component in TEs is controlled by CH. Please note the special role of the system component, which exists only conceptually and not as a running test component in a TE. System ports, i.e., the ports of the system component, may be distributed among several nodes. Further, test components on different nodes may have access to the same physical port of the SUT, i.e., they may be mapped to the same port of the test system interface.

Example: Access to remote real SUT ports can be realized by TEs via local proxies.

Communication between TTCN-3 components is either message- or procedure-based. Therefore, the CH adapts message and procedure-based communication of TTCN-3 components to the particular execution platform of the test system. It is aware of connections between TTCN-3 test component communication ports. It propagates send request operations from one TTCN-3 component to another TTCN-3 component. The receiving component may reside in a different instance of the same TE located on a different node. It then notifies the TE of any received test events by enqueueing them in the port queues of the TE.

Procedure-based communication operations between TTCN-3 components are also visible at the CH. The CH shall distinguish between the different kinds of procedure-based communication, i.e., call, reply, and exception, and shall propagate them in the appropriate manner to the TE where the target component resides. TTCN-3 procedure-based communication semantics, i.e., the effect of such operation on TTCN-3 test component execution, are to be handled in the TE.

Additional communication is needed to implement the distribution of test components onto several nodes. Component management communication includes the indication of the creation of test

components, the starting of execution of a test component, verdict distribution, as well as component termination indication. The CH does not implement the TTCN-3 component behaviour. Rather, it implements the communication between several components implemented by a TE.

6.1.1.4 Test logging (TL)

The TL entity performs test event logging and presentation to the test system user. It provides the logging of information about the test execution such as which test components have been created, started and terminated, which data is sent to the SUT, received from the SUT and matched to TTCN-3 templates, which timers have been started, stopped or timed out, etc.

6.1.2 TTCN-3 executable (TE)

The TE entity executes or interprets a TTCN-3 module. Conceptually, the TE can be decomposed into six interacting entities: a Control, Behaviour, Component, Type, Value, and Queue entity. This structural decomposition of the TE is defined in [ITU-T Z.165]. The terminology for TE defined in [ITU-T Z.165] is used within this Recommendation.

6.1.3 SUT adaptor (SA)

The SA is the implementation of the system under test adaptor (SA) as defined in [ITU-T Z.165]. The terminology for SA defined in [ITU-T Z.165] is used within this Recommendation.

6.1.4 Platform adaptor (PA)

The PA is the implementation of the platform adaptor (PA) as defined in [ITU-T Z.165]. The terminology for PA defined in [ITU-T Z.165] is used within this Recommendation.

6.2 Execution requirements for a TTCN-3 test system

Each TCI operation call shall be treated as an atomic operation in the calling entity. The called entity, which implements a TCI operation, shall return control to the calling entity as soon as its intended effect has been accomplished or if the operation cannot be completed successfully. The called entity shall not block in the implementation of procedure-based communication.

As stated before, no assumption is made as to whether the TTCN-3 test system or individual entities are implemented in a single executable or process or whether they are distributed among different processes or even test devices.

A TCI implementation shall fulfil the above-mentioned requirements.

7 TTCN-3 control interface and operations

This clause defines a set of abstract data types used to represent data communicated between the TE and the TMC. In addition, it defines TCI operations in terms of their signatures, when they are to be used and what their effects on the TTCN-3 test system are.

This definition also includes a more detailed description of the input parameters required for each TCI operation call and its return value.

7.1 Overview of the TCI

The TCI defines the interaction between the TTCN-3 executable (TE), component handling (CH), the test management (TM), the coding/decoding (CD), the test logging (TL) entities within a TTCN-3 test system. It provides means for the TE to:

• manage test execution;

• distribute execution of test components among different test devices;

• encode and decode test data; and

- logging of information about test execution.

The TCI consists of four sub-interfaces:

- *TCI test management interface (TCI-TM)*: This interface includes all operations needed to manage test execution, provide module parameters and external constants and provide test event logging.

- *TCI component handling interface (TCI-CH)*: This interface consists of operations needed to implement the management of, and communication between TTCN-3 test components in a centralized or distributed test system. It includes operations to create, start and stop test components, establish connection between TTCN-3 components, manage test components and their verdicts, and handle message and procedure-based communication between TTCN-3 components.

- *TCI coding/decoding interface (TCI-CD)*: This interface includes all operations needed to retrieve and access codecs, i.e., encoders or decoders, for encoding data to be sent, defined using the TTCN-3 encode attribute, and to decode received data.

- *TLI test logging interface (TCI-TL)*: This interface includes all operations needed to retrieve information about test execution and to control the level of detail of these information.

All interfaces are bidirectional so that calling and called parts reside in the TE and in the TMC of the test system. The provided interfaces (those operations which an interface offers to the TE) and the required operations (those operations which an interface needs to use from the TE) are combined into the respective provided and required subinterface for each interface, i.e., TCI-TM Provided/TCI-TM Required, TCI-CH Provided/TCI-CH Required, TCI-CD Provided/TCI-CD Required, and TCI-TL Provided/TCI-TL Required.

7.1.1 Correlation between TTCN-3 and TCI operation invocations

For some TTCN-3 operation invocations, there is a direct correlation to a TCI operation invocation, which is shown in Table 1. Some of the TTCN-3 operations correlate to a pair of TCI operation request and TCI operation to implement the propagation of TTCN-3 operations through the test system. For the other TCI operation invocations there is an indirect correlation – they are needed to implement the TTCN-3 semantics of underlying concepts.

The correlation shown for TTCN-3 communication operations (i.e., `send`, `call`, `reply`, and `raise`) only holds if these operations are invoked on a test component port connected to another test component port. The correlation for communication operations that are invoked on test component ports that are mapped to test system interface ports is defined in [ITU-T Z.165].

Table 1 – Correlation between TTCN-3 and TCI operation invocations

TTCN-3 operation name	TCI operation name	TCI interface name
`send`	`tciSendConnected` (see Note 1)	TCI-CH Provided
	`tciSendConnectedBC` (see Note 2)	
	`tciSendConnectedMC` (see Note 3)	
	`tciEnqueueMsgConnected`	TCI-CH Required
`call`	`tciCallConnected` (see Note 1)	TCI-CH Provided
	`tciCallConnectedBC` (see Note 2)	
	`tciCallConnectedMC` (see Note 3)	
	`tciEnqueueCallConnected`	TCI-CH Required

Table 1 – Correlation between TTCN-3 and TCI operation invocations

TTCN-3 operation name	TCI operation name	TCI interface name
reply	tciReplyConnected (see Note 1)	TCI-CH Provided
	tciReplyConnectedBC (see Note 2)	
	tciReplyConnectedMC (see Note 3)	
	tciEnqueueReplyConnected	TCI-CH Required
raise	tciRaiseConnected (see Note 1)	TCI-CH Provided
	tciRaiseConnectedBC (see Note 2)	
	tciRaiseConnectedMC (see Note 3)	
	tciEnqueueRaiseConnected	TCI-CH Required
create	tciCreateTestComponentReq	TCI-CH Provided
	tciCreateTestComponent	TCI-CH Required
start (a component)	tciStartTestComponentReq	TCI-CH Provided
	tciStartTestComponent	TCI-CH Required
stop (a component)	tciStopTestComponentReq	TCI-CH Provided
	tciStopTestComponent	TCI-CH Required
kill	tciKillTestComponentReq	TCI-CH Provided
	tciKillTestComponent	TCI-CH Required
connect	tciConnectReq	TCI-CH Provided
	tciConnect	TCI-CH Required
disconnect	tciDisconnectReq	TCI-CH Provided
	tciDisconnect	TCI-CH Required
map	tciMapReq	TCI-CH Provided
	tciMap	TCI-CH Required
unmap	tciUnmapReq	TCI-CH Provided
	tciUnmap	TCI-CH Required
running	tciTestComponentRunningReq	TCI-CH Provided
	tciTestComponentRunning	TCI-CH Required
alive	tciTestComponentAliveReq	TCI-CH Provided
	tciTestComponentAlive	TCI-CH Required
done	tciTestComponentDoneReq	TCI-CH Provided
	tciTestComponentDone	TCI-CH Required
killed	tciTestComponentKilledReq	TCI-CH Provided
	tciTestComponentKilled	TCI-CH Required
mtc	tciGetMTCReq	TCI-CH Provided
	tciGetMTC	TCI-CH Required
execute	tciTestCaseExecuteReq	TCI-CH Provided
	tciTestCaseExecute	TCI-CH Required

Table 1 – Correlation between TTCN-3 and TCI operation invocations

TTCN-3 operation name	TCI operation name	TCI interface name
`log`	`tliLog`	TCI-TL Provided
NOTE 1 – For unicast communication.		
NOTE 2 – For broadcast communication.		
NOTE 3 – For multicast communication.		

7.2 TCI data

The TCI specification defines a set of abstract data types. These describe, at a very high level, which kind of data shall be passed from a calling to a called entity. The abstract data types are used to determine:

• how TTCN-3 data is passed from a TE to an encoder, to encode TTCN-3 value representations into a bitstring; and in the reverse case;

• how data passed from a decoder to the TE shall be decoded from a bitstring into its TTCN-3 value representation.

For these abstract data types a set of operations is defined to process the data by the coder/decoder.

The concrete representation of these abstract data types as well as the definition of basic data types like `string` and `boolean` are defined in the respective language mappings in clauses 8 and 9.

Notice that the values for any identifier data type shall be unique in the test system implementation where uniqueness is defined as being globally distinct at any point in time. This guarantees that different objects, e.g., two timers, are identified by different identifiers and identifiers are not reused.

7.2.1 General abstract data types

The following abstract data types are defined and used for the definition of TCI operations.

7.2.1.1 Management

`TciModuleIdType` A value of `TciModuleIdType` is the name of a TTCN-3 module as specified in the TTCN-3 ATS. This abstract type is used for module handling.

`TciModuleParameterIdType` A value of `TciModuleParameterIdType` is the qualified name of a TTCN-3 module parameter as specified in the TTCN-3 ATS. This abstract type is used for module parameter handling.

`TciTestCaseIdType` A value of `TciTestCaseIdType` is the qualified name of a TTCN-3 testcase as specified in the TTCN-3 ATS. This abstract type is used for testcase handling.

`TciModuleIdListType` A value of type `TciModuleIdListType` is a list of `TciModuleIdType`. This abstract type is used when retrieving the list of modules which are imported by a TTCN-3 module.

`TciModuleParameterType` A value of type `TciModuleParameterType` is a structure of `TciModuleParameterIdType` and `Value`. This abstract type is used to represent the parameter name and the default value of a module parameter.

`TciModuleParameterListType`	A value of type `TciModuleParameterListType` is a list of `TciModuleParameterType`. This abstract type is used when retrieving the module parameters of a TTCN-3 module.
`TciParameterType`	A value of type `TciParameterType` includes a TTCN-3 `Value` and a value of `TciParameterPassingModeType` to represent the parameter passing mode specified for the parameter in the TTCN-3 ATS.
`TciParameterPassingModeType`	A value of type `TciParameterPassingModeType` is either `IN`, `INOUT`, or `OUT`. This abstract type is used when starting a test case or when the termination of a test case is indicated.
`TciParameterListType`	A value of type `TciParameterListType` is a list of `TciParameterType`. This abstract type is used when starting a test case or when the termination of a test case is indicated.
`TciParameterTypeType`	A value of type `TciParameterTypeType` is a structure of `Type` and `TciParameterPassingModeType`. This abstract type is used to represent the type and the parameter passing mode of a test case parameter.
`TciParameterTypeListType`	A value of type `TciParameterTypeListType` is a list of `TciParameterTypeType`. This abstract type is used to represent the list of parameters of a test case.
`TciTestComponentKindType`	A value of type `TciTestComponentKindType` is a literal of the set of kinds of TTCN-3 test components, i.e., `CONTROL`, `MTC`, `PTC`, `SYSTEM`, and `PTC_ALIVE`. This abstract type is used for component handling.
`TciTypeClassType`	A value of type `TciTypeClassType` is a literal of the set of type classes in TTCN-3 such as boolean, float, record, etc. This abstract type is used for value handling.

7.2.1.2   Communication

`TciBehaviourIdType`	A value of type `TciBehaviourIdType` identifies a TTCN-3 behaviour functions.

Additional abstract data types with the prefix Tri are taken from [ITU-T Z.165]: `TriPortIdType`, `TriPortIdListType`, `TriComponentIdType`, `TriComponentIdListType`, `TriAddressType`, `TriAddressListType`, and `TriMessageType`.

7.2.2   Abstract TTCN-3 data types and values

This clause defines the set of abstract data types that build up the TTCN-3 type and value representation. Functionality of each data type is defined by an accompanying set of operations. Operations on or using this abstract data type return either a value of this abstract type or a basic type like `boolean`.

All operations have been defined using the Interface Description Language (IDL). Concrete language mappings for the operations on the abstract data types are given in clauses 8 and 9. In certain languages, the application of an operation on an abstract data type is represented by passing (either by-value or by-reference, depending on the mapping) the concrete value as a parameter to the operation. Other languages might choose other referencing method to the concrete value, e.g., by considering the value as an object on which a method corresponding to the operation is invoked. To indicate the inability to perform a certain task or to indicate the absence of an optional parameter in the following, the distinct value `null` is used. It can be considered as being a reserved value

indicating a special value. The language mappings will define a concrete representation of this distinct value `null`.

The abstract TTCN-3 type and value representation consists of two parts:

• an abstract data type `Type` that represents all TTCN-3 types in a TTCN-3 module;

• different abstract data types that represent TTCN-3 values, i.e., TTCN-3 values of a given TTCN-3 type. This can be either values of TTCN-3 predefined types or of TTCN-3 user-defined types.

For accessing, evaluating, and coding the TTCN-3 data, the test system uses the abstract data type `Type` and the different abstract value data types. Therefore, these abstract data types define the abstraction level between the TTCN-3 Executable (TE) and the remaining test system using the TCI interfaces.

7.2.2.1   Abstract TTCN-3 data types

According to this Recommendation, TTCN-3 types, either predefined or user-defined, will be represented at the TCI interfaces using the abstract data type `Type`.

For the abstract data type `Type` a set of operations is defined to:

• reference predefined and user-defined TTCN-3 data types; and

• create and maintain TTCN-3 values.

The following operations are defined for the abstract data type `Type`:

`TciModuleIdType getDefiningModule()`	Returns the module identifier of the module in which type is defined. Returns the distinct value `null` if type is a TTCN-3 base type, e.g., boolean, integer, etc.
`TString getName()`	Returns the name of the type as defined in the TTCN-3 module.
`TciTypeClassType getTypeClass()`	Returns the type class of the respective type. A value of `TciTypeClassType` can have one of the following constants: `ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OBJID`, `OCTETSTRING`, `RECORD`, `RECORD_OF`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`, `VERDICT`.
`Value newInstance()`	Returns a freshly created value of the given type. This initial value of the created value is undefined.
`TString getTypeEncoding()`	Returns the type encoding attribute as defined in the TTCN-3 module.
`TString getTypeEncodingVariant()`	This operation returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute is defined the distinct value `null` is returned.
`TStringseq getTypeExtension()`	Returns the type extension attribute as defined in the TTCN-3 module.

7.2.2.2   Abstract TTCN-3 values

According to this Recommendation, TTCN-3 values are represented at the TCI interfaces via numerous abstract data types.

Figure 4 presents the hierarchy between the abstract data types for TTCN-3 values (short: abstract values).
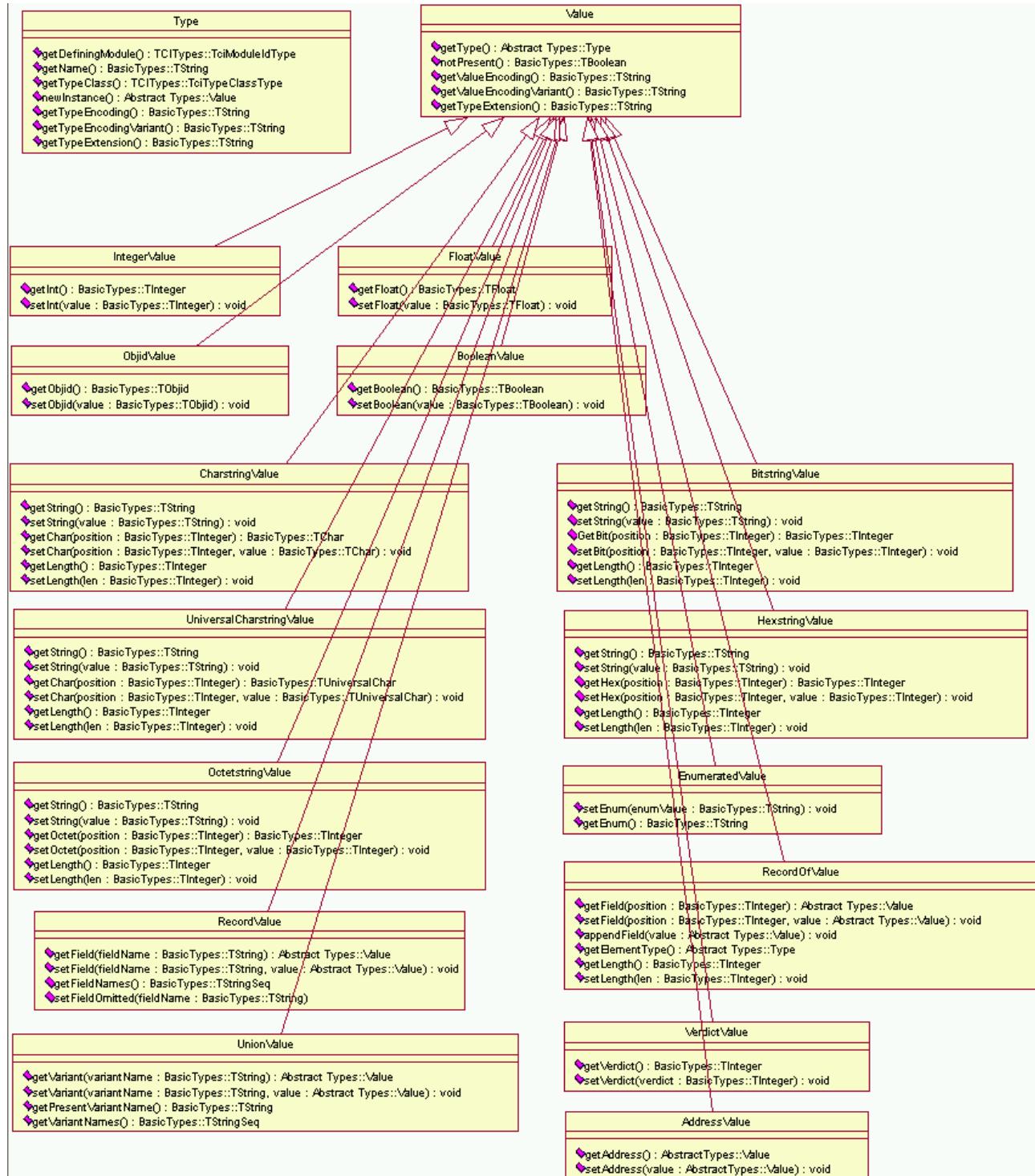
Figure 4 – Hierarchy of abstract values

As shown in Figure 4, all TTCN-3 abstract values share the same base abstract data type `Value`. All operations defined on this common base data type are implicitly defined also for the abstract value types derived from it.

7.2.2.2.1 The abstract data type `Value`

The following operations are defined on the base abstract data type `Value`. The concrete representations of these operations are defined in the respective language mapping sections:

`Type getType()`	Returns the type of the specified value.
`TBoolean notPresent()`	Returns `true` if the specified value is `omit`, `false` otherwise.
`TString getValueEncoding()`	Returns the value encoding attribute as defined in TTCN-3, if any. If no encoding attribute is defined the distinct value `null` is returned.
`TString getValueEncodingVariant()`	Returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute is defined the distinct value `null` is returned.

7.2.2.2.2 The abstract data type `IntegerValue`

The abstract data type `IntegerValue` is based on the abstract data type `Value`. It represents TTCN-3 `integer` values.

The following operations are defined on the abstract data type `IntegerValue`:

`TInteger getInt()`	Returns the integer value of this TTCN-3 integer.
`void setInt(in TInteger value)`	Sets this `IntegerValue` to `value`.

7.2.2.2.3 The abstract data type `FloatValue`

The abstract data type `FloatValue` is based on the abstract data type `Value`. It represents TTCN-3 `float` values.

The following operations are defined on the abstract data type `FloatValue`:

`TFloat getFloat()`	Returns the float value of this TTCN-3 float.
`void setFloat(in TFloat value)`	Sets this `FloatValue` to value.

7.2.2.2.4 The abstract data type `BooleanValue`

The abstract data type `BooleanValue` is based on the abstract data type `Value`. It represents TTCN-3 `boolean` values.

The following operations are defined on the abstract data type `FloatValue`:

`TBoolean getBoolean ()`	Returns the boolean value of the TTCN-3 `boolean`.
`void setBoolean(in TBoolean value)`	Sets this boolean value to `value`.

7.2.2.2.5 The abstract data type `ObjidValue`

The abstract data type `ObjidValue` is based on the abstract data type `Value`. It represents TTCN-3 `objid` values.

The following operations are defined on the abstract data type `ObjidValue`:

`TObjid getObjid()`	Returns the object id value of the TTCN-3 `objid`.
`void setObjid(in TObjid value)`	Sets this `ObjidValue` to `value`.

7.2.2.2.6 The abstract data type `CharstringValue`

The abstract data type `CharstringValue` is based on the abstract data type `Value`. It represents TTCN-3 `charstring` values. `TChar` is a character within a charstring value.

The following operations are defined on the abstract data type `CharstringValue`:

`TString getString()`	Returns the string value of the TTCN-3 `charstring`. The textual representation of the empty TTCN-3 `charstring` is `''`, while its length is zero.
`void setString(in TString value)`	Sets this `CharstringValue` to `value`.
`TChar getChar(in TInteger position)`	Returns the char value of the TTCN-3 charstring at position. Position 0 denotes the first char of the TTCN-3 `charstring`. Valid values for position are from `0` to `length – 1`.
`void setChar(in TInteger position, in TChar value)`	Set the character at position to value. Valid values for position are from `0` to `length – 1`.
`TInteger getLength()`	Returns the length of this `CharstringValue` in chars, zero if the value of this `CharstringValue` is omit.
`void setLength(in TInteger len)`	setLength first resets this `CharstringValue` to its initial value and afterwards sets the length of this `CharstringValue` in chars to `len`.

7.2.2.2.7 The abstract data type `UniversalCharstringValue`

The abstract data type `UniversalCharstringValue` is based on the abstract data type `Value`. It represents TTCN-3 `universal charstring` values. `TUniversalChar` is a character within a universal charstring value.

The following operations are defined on the abstract data type `UniversalCharstringValue`:

`TString getString()`	Returns the textual representation of this `UniversalCharstringValue`, as defined in TTCN-3.
`void setString(in TString value)`	Sets the value of this `UniversalCharstringValue` according to the textual representation as defined by `value`.
`TUniversalChar getChar(in TInteger position)`	Returns the universal char value of the TTCN-3 `universal charstring` at position. Position 0 denotes the first `TUniversalChar` of the TTCN-3 `universal charstring`. Valid values for `position` are from `0` to `length – 1`.
`void setChar(in TInteger position, in TUniversalChar value)`	Sets the universal char at position to value. Valid values for position are from `0` to `length – 1`.
`TInteger getLength()`	Returns the length of this universal charstring value in universal chars, zero if the value of this universal charstring value is `omit`.

`void setLength(in TInteger len)`	setLength first resets this `UniversalCharstringValue` to its initial value and afterwards sets the length of this `UniversalCharstringValue` in universal chars to `len`.

7.2.2.2.8 The abstract data type `BitstringValue`

The abstract data type `BitstringValue` is based on the abstract data type `Value`. It represents TTCN-3 `bitstring` values.

The following operations are defined on the abstract data type `BitstringValue`.

`TString getString()`	Returns the textual representation of this `BitstringValue`, as defined in TTCN-3. E.g., the textual representation of `0101` is `"0101"B`. The textual representation of the empty TTCN-3 `bitstring` is `""B`, while its length is zero.
`void setString(in TString value)`	Sets the value of this `BitstringValue` according to the textual representation as defined by `value`. E.g., the value of this `BitstringValue` is `0101` if the textual representation in `value` is `"0101"B`.
`TChar getBit(in TInteger position)`	Returns the value (`0` | `1`) at position of this TTCN-3 bitstring as a character. Position 0 denotes the first bit of the TTCN-3 bitstring. Valid values for position are from `0` to `length - 1`.
`void setBit(in TInteger position, in TInteger value)`	
	Sets the bit at position to the value (`0` | `1`). Position 0 denotes the first bit in this BitstringValue. Valid values for position are from `0` to `length - 1`.
`TInteger getLength()`	Returns the length of this `BitstringValue` in bits, zero if the value of this `BitstringValue` is `omit`.
`void setLength(in TInteger len)`	setLength first resets this `BitstringValue` to its initial value and afterwards sets the length of this `BitstringValue` in bits to `len`.

7.2.2.2.9 The abstract data type `OctetstringValue`

The abstract data type `OctetstringValue` is based on the abstract data type `Value`. It represents TTCN-3 `octetstring` values.

The following operations are defined on the abstract data type `OctetstringValue`:

`TString getString()`	Returns the textual representation of this `OctetstringValue`, as defined in TTCN-3. E.g., the textual representation of `0xCAFFEE` is `"CAFFEE"O`. The textual representation of the empty TTCN-3 `octetstring` is `""O`, while its length is zero.
`void setString(in TString value)`	Sets the value of this `OctetstringValue` according to the textual representation as defined by `value`. E.g., the value of this `OctetstringValue` is `0xCAFFEE` if the textual representation in `value` is `"CAFFEE"O`.

```
TChar getOctet(in TInteger position)
```
Returns the value `(0..255)` at position of this TTCN-3 octetstring. Position 0 denotes the first octet of the TTCN-3 octetstring. Valid values for position are from `0` to `length - 1`.

```
void setOctet(in TInteger position, in TInteger value)
```
Sets the octet at position to value `(0..255)`. Position 0 denotes the first octet in the octetstring. Valid values for position are from `0` to `length - 1`.

```
TInteger getLength()
```
Returns the length of this `OctetstringValue` in octets, zero if the value of this `OctetstringValue` is `omit`.

```
void setLength(in TInteger len)
```
setLength first resets this `OctetstringValue` to its initial value and afterwards sets the length of this `OctetstringValue` in octets to `len`.

7.2.2.2.10 The abstract data type `HexstringValue`

The abstract data type `HexstringValue` is based on the abstract data type `Value`. It represents TTCN-3 `hexstring` values.

The following operations are defined on the abstract data type `HexstringValue`:

```
TString getString()
```
Returns the textual representation of this `HexstringValue`, as defined in TTCN-3. E.g., the textual representation of `0xAFFEE` is `"AFFEE"H`. The textual representation of the empty TTCN-3 `hexstring` is `""H`, while its length is zero.

```
void setString(in TString value)
```
Sets the value of this `HexstringValue` according to the textual representation as defined by `value`. E.g., The value of this `HexstringValue` is `0xAFFEE` if the textual representation in `value` is `"AFFEE"H`.

```
TChar getHex(in TInteger position)
```
Returns the value `(0..15)` at position of this TTCN-3 hexstring. Position 0 denotes the first hex digits of the TTCN-3 hexstring. Valid values for position are from `0` to `length - 1`.

```
void setHex(in TInteger position, in TInteger value)
```
Sets the hex digit at position to value `(0..15)`. Position 0 denotes the first octet in the `hexstring`. Valid values for position are from `0` to `length - 1`.

```
TInteger getLength()
```
Returns the length of this `HexstringValue` in octets, zero if the value of this `HexstringValue` is `omit`.

```
void setLength(in TInteger len)
```
setLength first resets this `HexstringValue` to its initial value and afterwards sets the length of this `HexstringValue` in hex digits to `len`.

7.2.2.2.11 The abstract data type `RecordValue`

The abstract data type `RecordValue` is based on the abstract data type `Value`. It specifies how to get and set the TTCN-3 `record` type. The same abstract data type applies for values whose type class is `SET`. The distinction between `record` and `set` is only relevant at matching time.

The following operations are defined on the abstract data type `RecordValue`:

`Value getField(in TString fieldName)`

Returns the value of the field named `fieldName`. The return value is the common abstract base type `Value`, as a record field can have any type defined in TTCN-3. If the field cannot be obtained from the record the distinct value `null` is returned.

`void setField(in TString fieldName, in Value value)`

Sets the field named `fieldName` of the record to `value`. No assumption shall be made on how a field is stored in a record. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value is copied. Therefore, it should be assumed that subsequent modifications of `value` will not be considered in the record.

`TStringSeq getFieldNames()`

Returns a sequence of string of field names, the empty sequence, if the record has no fields.

`void setFieldOmitted(in TString fieldName)`

Mark the referenced field of the record as being omitted.

7.2.2.2.12 The abstract data type `RecordOfValue`

The abstract data type `RecordOfValue` is based on the abstract data type `Value`. It specifies how to get and set elements in TTCN-3 `record of` types. The same abstract data type applies for value whose type class is `SET_OF`. The distinction between `record of` and `set of` is only relevant at matching time.

The following operations are defined on the abstract data type `RecordOfValue`:

`Value getField(in TInteger position)`

Returns the value of the `record of` at `position` if position is between `zero` and `length - 1`, the distinct value `null` otherwise. The return value is the common abstract base type `Value`, as a `record of` can have fields of any type defined in TTCN-3.

`void setField(in TInteger position, in Value value)`

Sets the field at `position` to `value`. If `position` is greater than `(length - 1)` the record of is extended to have the length `(position + 1)`. The `record of` elements between the original position at `length` and `position - 1` is set to `omit`. No assumption shall be made on how a field is stored in a `record of`. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value is copied. Therefore, it should be assumed that subsequent modifications of `value` will not be considered in the `record of`.

void appendField(in Value value)	Appends the value at the end of the record of, i.e., at position length. No assumption shall be made on how a field is stored in a record of. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value is copied. Therefore, it should be assumed that subsequent modifications of value will not be considered in the record of.
Type getElementType()	Returns the Type of the elements of this record of.
TInteger getLength()	Returns the actual length of the record of value, zero if the record of value is omit.
void setLength(in TInteger len)	Sets the length of the record of to len. If len is greater than the original length, newly created elements have the value omit. If len is less than or equal to the original length, this operation is ignored.

7.2.2.2.13 The abstract data type `UnionValue`

The abstract data type UnionValue is based on the abstract data type Value. It specifies how to get and set variants in a TTCN-3 union type. The TTCN-3 anytype is represented by a UnionValue where the type class of the type obtained by getType() is ANYTYPE. For details on type classes see clause 7.2.2.1.

The following operations are defined on the abstract data type UnionValue:

Value getVariant(in TString variantName)	Returns the value of the TTCN-3 union variantName, if variantName equals the result of getPresentVariantName, the distinct value null otherwise. variantName denotes the name of the union variant as defined in TTCN-3.
void setVariant(in TString variantName, in Value value)	Sets variantName of the union to value. If variantName is not defined for this union, this operation is ignored. If another variant was selected, the new variant is selected instead.
TString getPresentVariantName()	Returns a String representing the currently selected variant name in the given TTCN-3 union. The distinct value null is returned if no variant is selected.
TStringSeq getVariantNames()	Returns a sequence of string of variant names, the distinct value null, if the union has no fields. If the UnionValue represents the TTCN-3 anytype, i.e., the type class of the type obtained by getType() is ANYTYPE, all predefined and user-defined TTCN-3 types is returned.

7.2.2.2.14 The abstract data type `EnumeratedValue`

The abstract data type EnumeratedValue is based on the abstract data type Value. It specifies how TTCN-3 enumerated can be set and get.

The following operations are defined on the abstract data type `EnumeratedValue`:

`TString getEnum()`	Returns the string identifier of this `EnumeratedValue`. This identifier equals the identifier in the TTCN-3 specification.
`void setEnum(in TString enumValue)`	Sets the enum to `enumValue`. If `enumValue` is not an allowed value for this enumeration, the operation is ignored.

7.2.2.2.15 The abstract data type `VerdictValue`

The abstract data type `VerdictValue` is based on the abstract data type `Value`. It specifies how TTCN-3 `verdict` can be set and get.

The following operations are defined on the abstract data type `VerdictValue`:

`TInteger getVerdict()`	Returns the integer value for this `VerdictValue`. The integer is one of the following constants: `ERROR`, `FAIL`, `INCONC`, `NONE`, `PASS`.
`void setVerdict(in TInteger verdict)`	Sets this `VerdictValue` to `verdict`. Note that a `VerdictValue` can be set to any of the above-mentioned verdicts at any time. The `VerdictValue` does not perform any verdict calculations as defined in TTCN-3. For example, it is legal to set the `VerdictValue` first to `ERROR` and then to `PASS`.

7.2.2.2.16 The abstract data type `AddressValue`

The following operations are defined on the base abstract data type `AddressValue`. The concrete representations of these operations are defined in the respective language mapping sections:

`Value getAddress()`	Returns the address value, which will no longer be of type class `ADDRESS` but rather of the actual type used for address.
`void setAddress(in Value value)`	Sets this address value to `value`.

7.2.3 Abstract logging types

7.2.3.1 The abstract data type `TciValueTemplate`

The following operations are defined on the abstract data type `TciValueTemplate`. The concrete representations of these operations are defined in the respective language mapping sections:

`boolean isOmit()`	Returns `true` if the template is an omit template.
`boolean isAny()`	Returns `true` if the template is an any template.
`boolean isAnyOrOmit()`	Returns `true` if the template is an any or omit template.
`TString getTemplateDef()`	Returns the definition of that template.

7.2.3.2 The abstract data type `TciNonValueTemplate`

The following operations are defined on the abstract data type `TciNonValueTemplate`. The concrete representations of these operations are defined in the respective language mapping sections:

`boolean isAny()`	Returns `true` if the template is an any template.

`boolean isAll()`	Returns `true` if the template is an all template.
`TString getTemplateDef()`	Returns the definition of that template.

7.2.3.3   The Value List and Mismatch Types

The following abstract data types are defined and used for the logging of differences between values and templates:

`TciValueList`	A value of `TciValueList` is a list of values.
`TciValueDifference`	A value of `TciValueDifference` is a structure containing a value, a template, and a description for the reason of this difference.
`TciValueDifferenceList`	A value of `TciValueDifferenceList` is a sequence of value differences.

7.2.3.4   The Status Types

The following abstract data types are defined and used for the logging of component and timer status:

`ComponentStatusType`	A value of `ComponentStatusType` is either "inactiveC", "runningC", "stoppedC", or "killedC".
`TimerStatusType`	A value of `TimerStatusType` is either "runningT", "inactiveT", or "expiredT".
`PortStatusType`	A value of `PortStatusType` is either "startedP", "haltedP", or "stoppedP".

7.3   TCI operations

This clause specifies the operations that a TTCN-3 Executable shall provide to a test system (*required operations*) and which functionality shall be provided by the test system to the TTCN-3 Executable (*provided operations*).

The terms "required" and "provided" reflect the fact that this Recommendation defines the requirements on a TTCN-3 Executable from a user's point of view. The user "requires" from a TTCN-3 Executable certain functionality to build a complete TTCN-3-based test system. To fulfil its task the TTCN-3 Executable has to inform the user on certain events where the user has to "provide" this possibility to the TTCN-3 Executable.

All operation definitions in this clause are defined using the interface definition language (IDL). Concrete language mappings are defined in clauses 8 and 9. Annex B provides for the logging interface an alternative mapping to XML.

For every TCI operation call, all *in*, *inout*, and *out* parameters listed in the particular operation definition are mandatory. The value of an *in* parameter is specified by the calling entity. Calling entity refers to the direction of the call. For operations on a *required* interface, the calling entity is the test system while the called entity is the TTCN-3 Executable. For operations on a *provided* interface, the calling entity is the TTCN-3 Executable while the test system is the called entity.

Similarly, the value of an *out* parameter is specified by the called entity. In the case of an *inout* parameter, a value is first specified by the calling entity but may be replaced with a new value by the called entity. Note that although TTCN-3 also uses *in*, *inout*, and *out* for signature definitions the denotations used in TCI IDL specification are not related to those in a TTCN-3 specification.

Operation calls should use a reserved value to indicate the absence of parameters. The reserved values for these types are defined in each language mapping and will be subsequently referred to as

the `null` value.

In addition, the `null` value will also be used to indicate the inability to perform a certain task.

As this clause specifies interfaces only and does not suggest concrete implementations on how to perform the specified functionality, the term entity will be used to identify the part of the test system implementation that implements this interface and performs the requested functionality. For example, the calling entity in the tciSendConnected operation is the TE, i.e., the part of test system implementation that provides the TE functionality.

All functions in the interface are described using the following template. Descriptions that are not applicable for certain operations are removed.

Signature	IDL Signature
In Parameters	Description of data passed as parameters to the operation from the calling entity to the called entity.
Out Parameters	Description of data passed as parameters to the operation from the called entity to the calling entity.
InOut Parameters	Description of data passed as parameters to the operation from the calling entity to the called entity and from the called entity back to the calling entity.
Return Value	Description of data returned from the operation to the calling entity.
Constraint	Description of any constraints when the operation can be called.
Effect	Behaviour required of the called entity before the operation may return.

7.3.1 The TCI-TM interface

The TCI test management interface (TCI-TM) describes the operations a TTCN-3 Executable is required to implement and the operations a test management implementation shall provide to the TE (Figure 5).
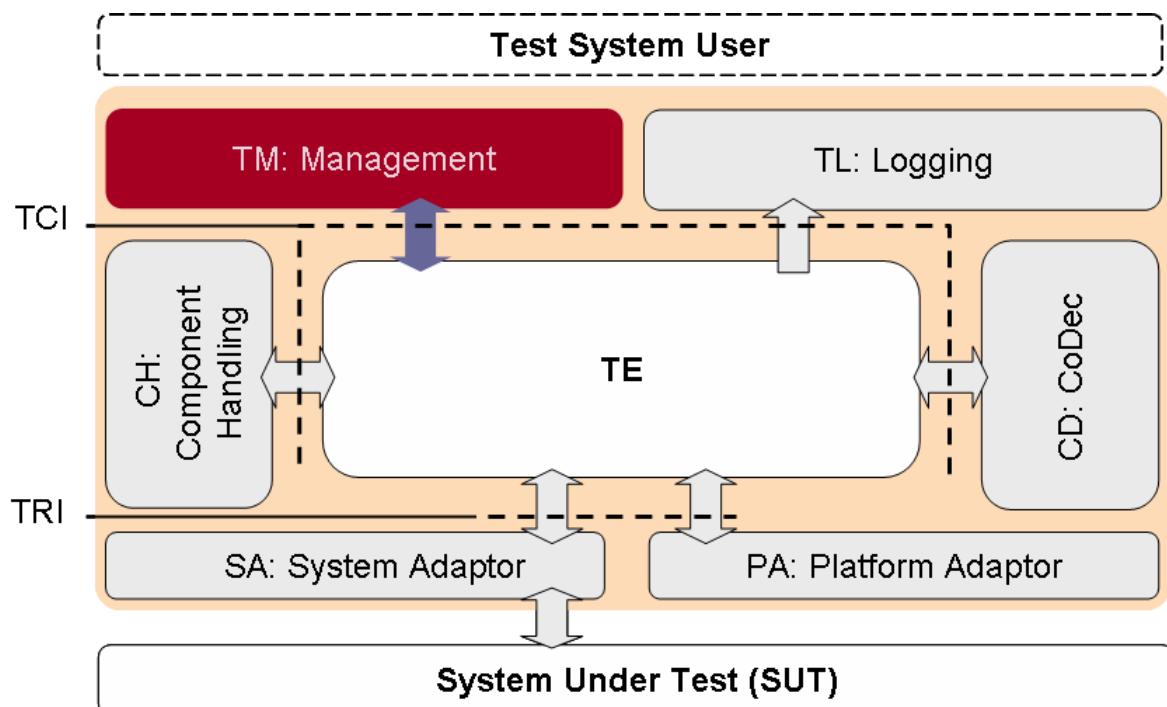
Figure 5 – The TCI-TM interface

A test management implementation provides overall test management to the test system user. It requires from the TE the presence of operations to start and stop test execution of a TTCN-3 module or of certain test cases in a TTCN-3 module. In turn it provides operations to the TE for resolving module parameter at runtime and the indication of execution termination.

Clause 10 illustrates the usage and sequential ordering of operation calls by either the TE or the test management.

7.3.1.1 TCI-TM required

This clause specifies the operations the TM requires from the TE. In addition to the operations specified in this clause, a test management requires the operations as required at the TCI-CD interface.

7.3.1.1.1 tciRootModule

Signature	void tciRootModule (in TciModuleIdType moduleName)	
In Parameters	moduleName	The moduleName denotes the module identifiers as defined in TTCN-3.
Return Value	void	
Constraint	Shall be used only if neither the control part nor a test case is currently being executed.	
Effect	tciRootModule selects the indicated module for execution through a subsequent call using tciStartTestCase or tciStartControl. A tciError will be issued by the TE if no such module exists.	

7.3.1.1.2 getImportedModules

Signature	TciModuleIdListType getImportedModules()
Return Value	A list of all imported modules of the root module. The modules are ordered as they appear in the TTCN-3 module. If no imported modules exist, an empty module list is returned.
Constraint	Shall be used only if a root module has been set before.
Effect	The TE provides to the management a list of imported modules of the root module. If no imported module exist, an empty module list is returned. If the TE cannot provide a list, the distinct null value shall be returned.

7.3.1.1.3 tciGetModuleParameters

Signature	TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName)	
In Parameters	moduleName	The moduleName denotes the module identifiers for which the module parameters should be retrieved.
Return Value	A list of all module parameters of the identified module. The parameters are ordered as they appear in the TTCN-3 module. If no parameters exist, an empty module parameter list is returned.	
Constraint	Shall be used only if a root module has been set before.	
Effect	The TE provides to the management a list of module parameters of the identified module. If no module parameters exist, an empty module parameter list is returned. If the TE cannot provide a list, the distinct null value shall be returned.	

7.3.1.1.4 tciGetTestCases

Signature	`TciTestCaseIdListType tciGetTestCases ()`
Return Value	A list of all test cases that are either defined in or imported into the root module.
Constraint	Shall be used only if a root module has been set before.
Effect	The TE provides to the management a list of test cases. If no test cases exist, an empty test case list is returned. If the TE cannot provide a list, the distinct `null` value shall be returned.

7.3.1.1.5 tciGetTestCaseParameters

Signature	`TciParameterTypeListType tciGetTestCaseParameters (in TciTestCaseIdType testCaseId)`	
In Parameters	`testCaseId`	A test case identifier as defined in the TTCN-3 module.
Return Value	A list of all parameter types of the given test case. The parameter types are ordered as they appear in the TTCN-3 signature of the test case. If no parameters exist, an empty parameter type list is returned.	
Constraint	Shall be used only if a root module has been set before.	
Effect	The TE provides to the management a list of parameter types of the given test case. If no test case parameters exist, an empty parameter type list is returned. If the TE cannot provide a list, the distinct `null` value shall be returned.	

7.3.1.1.6 tciGetTestCaseTSI

Signature	`TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)`	
In Parameters	`testCaseId`	A test case identifier as defined in the TTCN-3 module.
Return Value	A list of all system ports of the given test case that have been declared in the definition of the system component for the test case, i.e., the TSI ports. If a system component has not been explicitly defined for the test case, then the list contains all communication ports of the MTC test component. The ports are ordered as they appear in the respective TTCN-3 component type declaration. If no system ports exist, an empty list, i.e., a list of length zero is returned.	
Constraint	Shall be used only if a root module has been set before.	
Effect	The TE provides to the management a list of system ports of the given test case. If no system ports exist, an empty port list is returned. If the TE cannot provide a list, the distinct `null` value shall be returned.	

7.3.1.1.7 tciStartTestCase

Signature	void tciStartTestCase(in TciTestCaseIdType testCaseId, in TciParameterListType parameterList)	
In Parameters	`testCaseId`	A test case identifier as defined in the TTCN-3 module.
	`parameterList`	A list of values where each value defines a parameter from the parameter list as defined in the TTCN-3 test case definition. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature of the test case. If no parameters have to be passed either the `null` value or an empty `parameterList`, i.e., a list of length zero shall be passed.
Return Value	`void`	
Constraint	Shall be called only if a module has been selected before. Only `testCaseIds` for test cases that are declared in the currently selected TTCN-3 module shall be passed. Test cases that are imported in a referenced module cannot be started. To start imported test cases, the referenced (imported) module must be selected first using the `tciRootModule` operation.	
Effect	`tciStartTestCase` starts a testcase in the currently selected module with the given parameters. A `tciError` will be issued by the TE if no such test case exists. All *in* and *inout* test case parameters in `parameterList` contain `Value`. All *out* test case parameters in `parameterList` shall contain the distinct value of `null` since they are only of relevance when the test case terminates.	

7.3.1.1.8 tciStopTestCase

Signature	void tciStopTestCase()
Return Value	`void`
Constraint	Shall be called only if a module has been selected before.
Effect	`tciStopTestCase` stops the testcase currently being executed. If the TE is not executing a test case, the operation will be ignored. If the control part is being executed, `tciStopTestCase` will stop execution of the currently executed test case, i.e., the execution of the test case that has recently been indicated using the *provided* operation `tciTestCaseStarted`. A possible executing control part will continue execution as if the test case has stopped normally and returned with verdict `ERROR`.

7.3.1.1.9 tciStartControl

Signature	TriComponentId tciStartControl()
Return Value	A `TriComponentId` that represents the test component the module control part is executed on. If the TE cannot start control part of the selected module, the distinct value `null` will be returned.
Constraint	Shall be called only if a module has been selected before.
Effect	Starts the control part of the selected module. The control part will start TTCN-3 test cases as described in TTCN-3. While executing the control part, the TE will call the *provided* operations `tciTestCaseStarted` and `tciTestCaseTerminated` for every test case that has been started and that has terminated. After termination of the control part the TE will call the *provided* operation `tciControlPartTerminated`.

7.3.1.1.10 tciStopControl

Signature	`void tciStopControl()`
Return Value	`void`
Constraint	Shall only be called if a module has been selected before.
Effect	`tciStopControl` stops execution of the control part. If no control part is currently being executed the operation will be ignored. If a test case has been started directly this will stop execution of the current test case as if `tciStopTestCase` has been called.

7.3.1.2   TCI-TM provided

This clause specifies the operations the TM has to provide to the TE.

7.3.1.2.1 tciTestCaseStarted

Signature	`void tciTestCaseStarted(in TciTestCaseIdType testCaseId,` `in TciParameterListType parameterList,` `in TFloat timer)`	
In Parameters	`testCaseId`	A test case identifier as defined in the TTCN-3 module.
	`parameterList`	A list of values that are part of the test case signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 test case declaration.
	`timer`	A float value representing the duration of the test case timer.
Return Value	`void`	
Constraint	Shall only be called after either the control part of the module or a test case has been started using the *required* operations `tciStartControl` or `tciStartTestCase`.	
Effect	`tciTestCaseStarted` indicates to the TM that a test case with `testCaseId` has been started. It will not be distinguished whether the test case has been started explicitly using the *required* operation `tciStartTestCase` or implicitly while executing the control part.	

7.3.1.2.2 tciTestCaseTerminated

Signature	`void tciTestCaseTerminated(in VerdictValue verdict,` `in TciParameterListType parameterList)`	
In Parameters	`verdict`	The final verdict of the test case.
	`parameterList`	A list of values that are part of the test case signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 test case declaration.
Return Value	`void`	
Constraint	Shall only be called after either the control part of the module or a test case has been started using the *required* operations `tciStartControl` or `tciStartTestCase`.	
Effect	This operation will be called by the TE to indicate the test management that the test case that has been currently executed on the MTC has terminated and that the final verdict was `verdict`. On the invocation of a `tciTestCaseTerminated` operation all *out* and *inout* test case parameters contain `Values`. All *in* test case parameters contain the distinct value of `null` because they are only of relevance to the test case start but not in the reply to the call.	

7.3.1.2.3 tciControlTerminated

Signature	void tciControlTerminated ()
Return Value	void
Constraint	Shall only be called when the module execution has been started using the tciStartControl operation.
Effect	This operation will be called by the TE to indicate the test management that the control part of the selected module has just terminated execution.

7.3.1.2.4 tciGetModulePar

Signature	Value tciGetModulePar (in TciModuleParameterIdType parameterId)	
In Parameters	parameterId	The identifier of the module parameter as defined in TTCN-3.
Return Value	A value.	
Constraint	This operation shall be called whenever the TE needs to access the value of a module parameter. Every accessed module parameter will be resolved only once between a tciStartTestCase and tciTestCaseTerminated pair if a test case has been started explicitly or between a tciStartControl and tciControlTerminated pair if the control part of a module has been started.	
Effect	The management provides to the TE a Value for the indicated parameterId. Every call of tciGetModulePar() will return the same value throughout the execution of an explicitly started test case or throughout the execution of a control part. If the management cannot provide a TTCN-3 value, the distinct null value shall be returned.	

7.3.1.2.5 tciError

Signature	void tciError(in TString message)	
In Parameters	message	A string value, i.e., the message error
Return Value	void	
Constraint	Can be called at any time by the TE to indicate an unrecoverable error situation. This error situation could either be indicated by the CH or the CD or could occur within the TE.	
Effect	The TE indicates the occurrence of an unrecoverable error situation. message contains a reason phrase that might be communicated to the test system user. It is up to the test management to terminate execution of test cases or control parts if running. The test management has to take explicit measures to terminate test execution immediately.	

7.3.2 The TCI-CD interface

The TCI codec interface (TCI-CD) describes the operations a TTCN-3 Executable is required to implement and the operations a codec implementation for a certain encoding scheme shall provide to the TE (Figure 6).
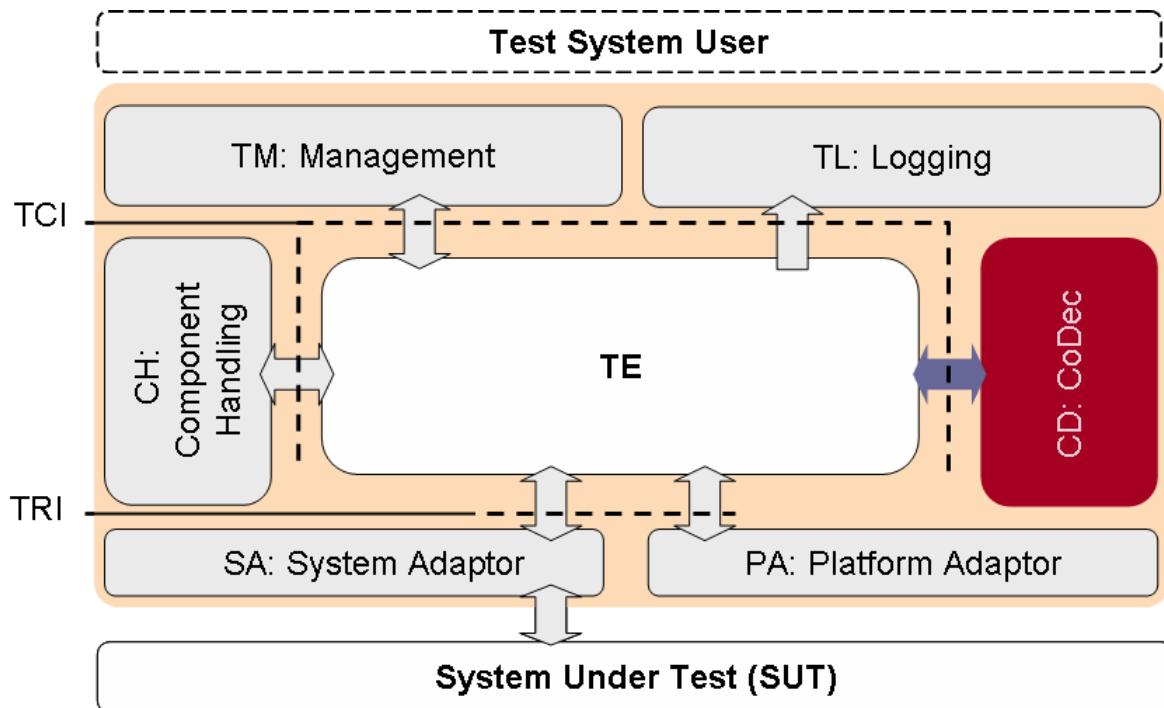
Figure 6 – The TCI-CD interface

A codec implementation encodes TTCN-3 values according to the encoding attribute into a bitstring and decodes a bitstring according to decoding hypothesis. To be able to decode a bitstring into a TTCN-3 value the CD requires certain functionality from the TE. In turn the CD provides encoding and decoding functionality to the TTCN-3 Executable.

Clause 10 illustrates the usage and sequential ordering of operation calls by either the TE or the CD.

7.3.2.1 TCI-CD required

This clause specifies the operations the CD requires from the TE. All operations specified in this clause are also required at the TCI-TM and TCI-CH interfaces.

7.3.2.1.1 getTypeForName

Signature	Type getTypeForName(in TString typeName)	
In Parameters	typeName	The TTCN-3 name of the type as defined in the TTCN-3 module. The following are reserved type names and will return a predefined type: "integer" "float" "bitstring" "hexstring" "octetstring" "charstring" "universal charstring" "boolean" "verdicttype" "objid" typeName has to be the fully qualified type name, i.e., module.typeName
Return Value	A type representing the requested TTCN-3 type.	
Constraint	–	
Effect	Returns a type representing a TTCN-3 type. Predefined TTCN-3 types can be retrieved from the TE by using the TTCN-3 keywords for the predefined types. In this case typeName denotes to the basic TTCN-3 type like "charstring", "bitstring" etc. Returns the distinct value null if the requested type cannot be returned. Note that the anytype and address cannot be obtained with module set to null. Although they are predefined types they might be distinct between modules. For example, address can either be the unmodified predefined type, or a user-defined type in a module. Other predefined types cannot be redefined.	

7.3.2.1.2 getInteger

Signature	Type getInteger()
Return Value	An instance of Type representing a TTCN-3 integer type.
Effect	Constructs and returns a basic TTCN-3 integer type.

7.3.2.1.3 getFloat

Signature	Type getFloat()
Return Value	An instance of Type representing a TTCN-3 float type.
Effect	Constructs and returns a basic TTCN-3 float type.

7.3.2.1.4 getBoolean

Signature	Type getBoolean()
Return Value	An instance of Type representing a TTCN-3 boolean type.
Effect	Constructs and returns a basic TTCN-3 boolean type.

7.3.2.1.5 getObjid

Signature	`Type getObjid()`
Return Value	An instance of `Type` representing a TTCN-3 object id type.
Effect	Constructs and returns a basic TTCN-3 object id type.

7.3.2.1.6 getCharstring

Signature	`Type getCharstring ()`
Return Value	An instance of `Type` representing a TTCN-3 charstring type.
Effect	Constructs and returns a basic TTCN-3 charstring type.

7.3.2.1.7 getUniversalCharstring

Signature	`Type getUniversalCharstring ()`
Return Value	An instance of `Type` representing a TTCN-3 universal charstring type.
Effect	Constructs and returns a basic TTCN-3 universal charstring type.

7.3.2.1.8 getHexstring

Signature	`Type getHexstring ()`
Return Value	An instance of `Type` representing a TTCN-3 hexstring type.
Effect	Constructs and returns a basic TTCN-3 hexstring type.

7.3.2.1.9 getBitstring

Signature	`Type getBitstring()`
Return Value	An instance of `Type` representing a TTCN-3 bitstring type.
Effect	Constructs and returns a basic TTCN-3 bitstring type.

7.3.2.1.10 getOctetstring

Signature	`Type getOctetstring ()`
Return Value	An instance of `Type` representing a TTCN-3 octetstring type.
Effect	Constructs and returns a basic TTCN-3 octetstring type.

7.3.2.1.11 getVerdict

Signature	`Type getVerdict()`
Return Value	An instance of `Type` representing a TTCN-3 verdict type.
Effect	Constructs and returns a basic TTCN-3 verdict type.

7.3.2.1.12 tciErrorReq

Signature	`void tciErrorReq(in TString message)`	
In Parameters	`Message`	A string value, i.e., the error phrase describing the problem.
Return Value	`void`	
Constraint	Shall be called whenever an error situation has occurred.	
Effect	The TE will be notified about an unrecoverable error situation within the CD and forward the error indication to the test management.	

7.3.2.2   TCI-CD provided

This clause specifies the operations the TM shall provide to the TE.

7.3.2.2.1 decode

Signature	`Value decode(in TriMessageType message,` `            in Type decodingHypothesis)`	
In Parameters	`message`	The encoded message to be decoded.
	`decodingHypothesis`	The hypothesis the decoding can be based on.
Return Value	Returns the decoded value, if the value is of a compatible type as the `decodingHypothesis`, else the distinct value `null`.	
Constraint	This operation shall be called whenever the TE has to decode an encoded value. The TE might decode immediately after reception of an encoded value, or might for performance considerations postpone the decoding until the actual access of the encoded value.	
Effect	This operation decode `message` according to the encoding rules and returns a TTCN-3 value. The `decodingHypothesis` shall be used to determine whether the encoded value can be decoded. If an encoding rule is not self-sufficient, i.e., if the encoded message does not inherently contain its type `decodingHypothesis` shall be used. If the encoded value can be decoded without the decoding hypothesis, the distinct `null` value shall be returned if the type determined from the encoded message is not compatible with the decoding hypothesis.	

7.3.2.2.2 encode

Signature	`TriMessageType encode(in Value value)`	
In Parameters	`value`	The value to be encoded.
Return Value	Returns an encoded `TriMessage` for the specified encoding rule.	
Constraint	This operation shall be called whenever the TE has to encode a Value.	
Effect	Returns an encoded `TriMessage` according to the encoding rules.	

7.3.3 The TCI-CH interface

The TCI component handling interface (TCI-CH) describes the operations a TTCN-3 Executable is required to implement and the operations a component handling implementation shall provide to the TE (Figure 7).
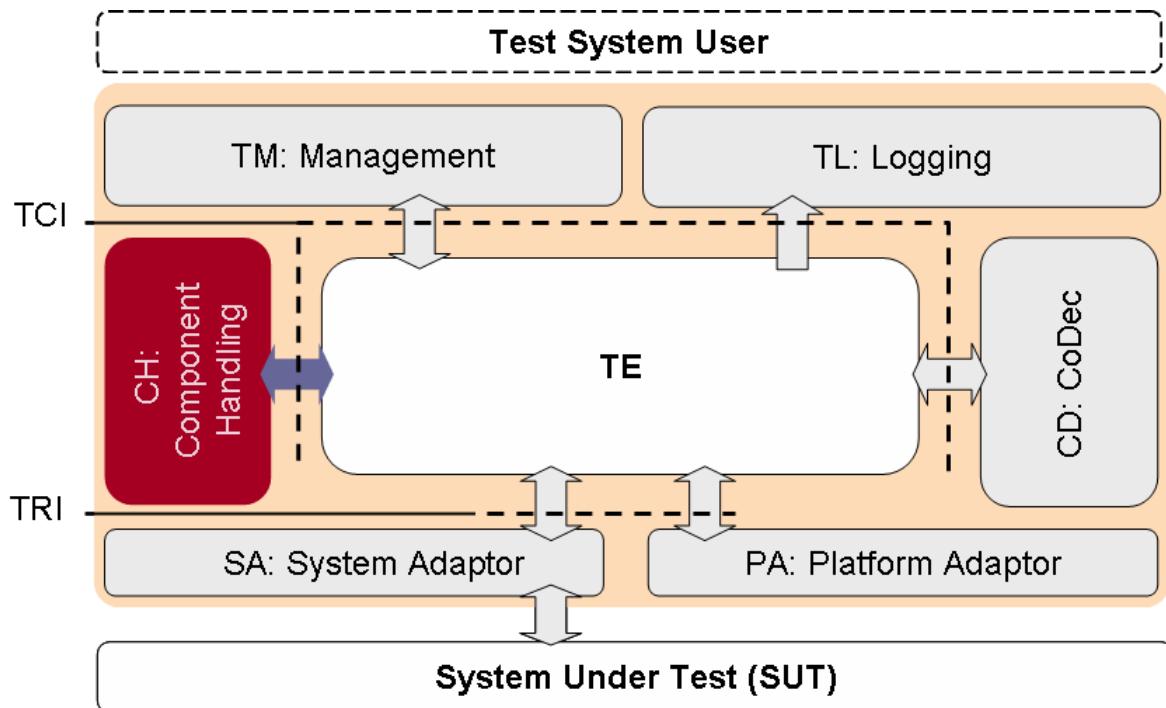
Figure 7 – The TCI-CH interface

A component handling implementation distributes TTCN-3 configuration operations like `create`, `connect` and `start` and intercomponent communication like `send` on a connected port among one or more TTCN-3 Executables participating in a test session. Note that although multiple instances of a TE might participate in a test session this is not mandatory.

The basic principle is that TCI-CH is not *implementing* any kind of TTCN-3 functionality. Instead it will be informed by the TE that for example a test component shall be created. Based on component handling (CH) internal knowledge, the request for creation of a test component will be transmitted to another (remote) participating TE. This second (remote) participating TE will create the TTCN-3 component and will provide a handle back to the requesting (local) TE. The requesting (local) TE can now operate on the created test component via this component handle.

Within the operation definitions, the terms "local TE" and "remote TE" are used to highlight the fact that a test system implementation might be distributed over several test devices, each of them hosting a complete TE. The terms "local" and "remote" always refer to the interfaces currently being described. For convenience, the term "local" refers always to the TE being either the callee of an operation (for *required* operations) or the caller of an operation (for *provided* operations). While the TE is conceptually considered as being distributed, the CH is considered to be non-distributed. This can either be achieved using a centralized architecture or by using a middleware-platform that abstracts from distribution aspects. Although the TE might be distributed over different physical devices, there might be configurations where only one, non-distributed TE will participate in a test session. In this case, the terms "local" and "remote" refer to the same TE instance.

Clause 10 illustrates the usage and sequential ordering of operation calls by either the TE or the CH.

Although all TTCN-3 Executables participating in a test session are equal, there is a distinct TE*. This TE* is the TE where the explicit `tciStartTestCase()` or `tciStartControl()` has been processed. The reason for this distinction is, that TE* shall calculate the global verdict. TE* will notify the test management upon termination of test execution and then shall provide the global verdict of the test case.

7.3.3.1 TCI-CH required

This clause specifies the operations the CH requires from the TE. In addition to the operations specified in this clause, all *required* operations of the TCI-CD interface are also required.

7.3.3.1.1 tciEnqueueMsgConnected

Signature	`void tciEnqueueMsgConnected (in TriPortIdType sender,`
	`                              in TriComponentIdType receiver,`
	`                              in Value rcvdMessage)`	

In Parameters	`sender`	Port identifier at the sending component via which the message is sent.
	`receiver`	Identifier of the receiving component.
	`rcvdMessage`	The value to be enqueued.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at remote TE a *provided* `tciSendConnected` has been called.	
Effect	The TE enqueues the received value into the local port queue of the indicated receiver component.	

7.3.3.1.2 tciEnqueueCallConnected

Signature	`void tciEnqueueCallConnected (in TriPortIdType sender,`
	`                               in TriComponentIdType receiver,`
	`                               in TriSignatureIdType signature,`
	`                               in TciParameterListType parameterList)`	

In Parameters	`sender`	Port identifier at the sending component via which the message is sent.
	`receiver`	Identifier of the receiving component.
	`signature`	Identifier of the signature of the procedure call.
	`parameterList`	A list of value parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciCallConnected` has been called. All *in* and *inout* procedure parameters contain values. All *out* procedure parameters shall contain the distinct value of `null` because they are only of relevance in a reply to the procedure call but not in the procedure call itself. The procedure parameters are the parameters specified in the TTCN-3 signature template.	
Effect	The TE enqueues the calls at the local port queue of the indicated receiver component.	

7.3.3.1.3 tciEnqueueReplyConnected

Signature	`void tciEnqueueReplyConnected (in TriPortIdType sender,`
	`                                in TriComponentIdType receiver,`
	`                                in TriSignatureIdType signature,`
	`                                in TciParameterListType`
	`                                parameterList,`
	`                                in Value returnValue)`	

In Parameters	`sender`	Identifier of the port sending the reply.
	`receiver`	Identifier of the component receiving the reply.
	`signature`	Identifier of the signature of the procedure call.
	`parameterList`	A list of value parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
	`returnValue`	(Optional) return value of the procedure call.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciReplyConnected` has been called. All *out* and *inout* procedure parameters and the return value contain values. All *in* procedure parameters shall contain the distinct value of `null` since they are only of relevance to the procedure call but not in the reply to the call. The `parameterList` contains procedure call parameters. These parameters are the parameters specified in the TTCN-3 signature template. If no return type has been defined for the procedure signature in the TTCN-3 ATS, the distinct value `null` shall be passed for the `returnValue`.	
Effect	The TE enqueues the reply at the local port queue of the indicated receiver component.	

7.3.3.1.4 tciEnqueueRaiseConnected

Signature	`void tciEnqueueRaiseConnected (in TriPortIdType sender,`
	`                                in TriComponentIdType receiver,`
	`                                in TriSignatureIdType signature,`
	`                                in Value exception)`	

In Parameters	`sender`	Identifier of the port sending the reply.
	`receiver`	Identifier of the component receiving the reply.
	`signature`	Identifier of the signature of the procedure call.
	`exception`	The exception.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciRaiseConnected` has been called.	
Effect	The TE enqueues the exception at the local port queue of the indicated receiver component.	

7.3.3.1.5 tciCreateTestComponent

Signature	`TriComponentIdType tciCreateTestComponent`
	`                         (in TciTestComponentKindType kind,`
	`                          in Type componentType),`
	`                          in TString name)`	

In Parameters	`kind`	The kind of component that shall be created, either `MTC`, `PTC` or `CONTROL`.
	`componentType`	Identifier of the TTCN-3 component type that shall be created.
	`name`	Name of the component that shall be created.
Return Value	A `TriComponentIdType` value for the created component.	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciCreateTestComponentReq` has been called. `componentType` shall be set to the distinct value `null` if a test component of kind `control` shall be created. `name` shall be set to the distinct value `null` if no name is given in the TTCN-3 create statement.	
Effect	The TE creates a TTCN-3 test component of the `componentType` and passes a `TriComponentIdType` reference back to the CH. The CH communicates the reference back to the remote TE.	

7.3.3.1.6 tciStartTestComponent

Signature	`void tciStartTestComponent(in TriComponentIdType component,`
	`                            in TciBehaviourIdType behaviour,`
	`                            in TciParamaterListType parameterList)`	

In Parameters	`component`	Identifier of the component to be started. Refers to an identifier previously created by a call of `tciCreateTestComponent`
	`behaviour`	Identifier of the behaviour to be started on the component.
	`parameterList`	A list of Values where each value defines a parameter from the parameter list as defined in the TTCN-3 function declaration of the function being started. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature of the test case. If no parameters have to be passed either the `null` value or an empty `parameterList`, i.e., a list of length zero shall be passed.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciStartTestComponentReq` has been called. Since only *in* parameters are allowed for functions being started [ITU-T Z.161], `parameterList` contains only *in* parameters.	
Effect	The TE shall start the indicated behaviour on the indicated component.	

7.3.3.1.7 tciStopTestComponent

Signature	`void tciStopTestComponent(in TriComponentIdType component)`	
In Parameters	`component`	Identifier of the component to be stopped.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciStopTestComponentReq` has been called.	
Effect	The TE shall stop the indicated behaviour on the indicated component.	

7.3.3.1.8 tciConnect

Signature	`void tciConnect (in TriPortIdType fromPort,` `                  in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be connected from.
	`toPort`	Identifier of the test component port to be connected to.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciConnect` has been called.	
Effect	The TE shall connect the indicated ports to one another.	

7.3.3.1.9 tciDisconnect

Signature	`void tciDisconnect (in TriPortIdType fromPort,` `                     in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be disconnected.
	`toPort`	Identifier of the test component port to be disconnected.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciDisconnect` has been called.	
Effect	The TE shall disconnect the indicated ports.	

7.3.3.1.10 tciMap

Signature	`void tciMap (in TriPortIdType fromPort,` `              in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be mapped from.
	`toPort`	Identifier of the test component port to be mapped to.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciMapReq` has been called.	
Effect	The TE shall map the indicated ports to one another.	

7.3.3.1.11 tciUnmap

Signature	void tciUnmap (in TriPortIdType fromPort, in TriPortIdType toPort)	
In Parameters	`fromPort`	Identifier of the test component port to be unmapped.
	`toPort`	Identifier of the test component port to be unmapped.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciUnmapReq` has been called.	
Effect	The TE shall unmap the indicated ports.	

7.3.3.1.12 tciTestComponentTerminated

Signature	void tciTestComponentTerminated
	                              (in TriComponentIdType component,
	                               in VerdictValue verdict)	

In Parameters	`component`	Identifier of the component that has terminated.
	`verdict`	Verdict after termination of the component.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciTestComponentTerminatedReq` has been called.	
Effect	The local TE is notified of the termination of the indicated test component on a remote TE. Since a function being executed on a test component can only have *in* parameters ([ITU-T Z.161]), the `tciTestComponentTerminated` operation does not have a `parameterList` parameter.	

7.3.3.1.13 tciTestComponentRunning

Signature	TBoolean tciTestComponentRunning (in TriComponentIdType component)	
In Parameters	`component`	Identifier of the component to be checked for running.
Return Value	`true` if the indicated component is still executing a behaviour, `false` otherwise.	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciTestComponentRunningReq` has been called.	
Effect	The local TE determines whether the indicated component is executing a test behaviour. If the component is executing a behaviour `true` will be returned. In any other case, e.g., test component has finished execution, or test component has not been started, etc. `false` will be returned. After the operation returns, the CH will communicate the value back to the remote TE.	

7.3.3.1.14   tciTestComponentDone

Signature	`TBoolean tciTestComponentDone (in TriComponentIdType comp)`
In Parameters	`comp` — Identifier of the component to be checked for done.
Return Value	`true` if the indicated component has completed executing its behaviour, `false` otherwise.
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciTestComponentDoneReq` has been called.
Effect	The local TE determines whether the indicated component has completed executing its test behaviour. If the component has completed its behaviour `true` will be returned. In any other case, e.g., test component has not been started, or test component is still executing, `false` will be returned. After the operation returns, the CH will communicate the value back to the remote TE.

7.3.3.1.15 tciGetMTC

Signature	`TriComponentIdType tciGetMTC()`
Return Value	A `TriComponentIdType` value of the MTC if the MTC executes on the local TE, the distinct value `null` otherwise.
Constraint	This operation can be called by the CH at the appropriate local TE when at a remote TE a *provided* `tciGetMTCReq` has been called.
Effect	The local TE determines whether the MTC is executing on the local TE. If the MTC executes on the local TE the component id of the MTC is being returned. If the MTC is not executed on the local TE the distinct value `null` will be returned. The operation will have no effect on the execution of the MTC. After the operation returns, the CH will communicate the value back to the remote TE.

7.3.3.1.16 tciExecuteTestCase

Signature	`void tciExecuteTestCase (in TciTestCaseIdType testCaseId,` `                          in TriPortIdListType tsiPortList)`
In Parameters	`testCaseId` — A test case identifier as defined in the TTCN-3 module.
	`tsiPortList` — Contains all ports that have been declared in the definition of the system component for the test case, i.e., the TSI ports. If a system component has not been explicitly defined for the test case, then the `tsiPortList` contains all communication ports of the MTC. The ports in `tsiPortList` are ordered as they appear in the respective TTCN-3 component type declaration. If no ports have to be passed either the `null` value or an empty `tsiPortList`, i.e., a list of length zero shall be passed.
Return Value	`void`
Constraint	This operation shall be called by the CH at the appropriate local TE when at a remote TE a *provided* `tciExecuteTestCaseReq` has been called.
Effect	The local TE determines whether static connections to the SUT and the initialization of communication means for TSI ports should be done.

7.3.3.1.17 tciReset

Signature	`void tciReset ()`
Return Value	`void`
Constraint	This operation shall be called by the CH at appropriate local TEs when at a remote TE a *provided* `tciResetReq` has been called.
Effect	The TE can decide to take any means to reset the test system locally.

7.3.3.1.18 tciKillTestComponent

Signature	`void tciKillTestComponent(in TriComponentIdType comp)`	
In Parameters	`comp`	Identifier of the component to be killed.
Return Value	`void`	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciKillTestComponentReq` has been called.	
Effect	The TE stops the behaviour on the indicated component if necessary and transfers it into the killed state.	

7.3.3.1.19 tciTestComponentAlive

Signature	`TBoolean tciTestComponentAlive (in TriComponentIdType comp)`	
In Parameters	`comp`	Identifier of the component to be checked for being alive.
Return Value	`true` if the indicated component is alive, `false` otherwise.	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciTestComponentAliveReq` has been called.	
Effect	The local TE determines whether the indicated component is alive. After the operation returns, the CH will communicate the value back to the remote TE.	

7.3.3.1.20 tciTestComponentKilled

Signature	`TBoolean tciTestComponentKilled (in TriComponentIdType comp)`	
In Parameters	`comp`	Identifier of the component to be checked for being killed.
Return Value	`true` if the indicated component has been killed, `false` otherwise.	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a *provided* `tciTestComponentKilledReq` has been called.	
Effect	The local TE determines whether the indicated component is in the killed state. If it is, `true` will be returned. In any other case, `false` will be returned. After the operation returns, the CH will communicate the value back to the remote TE.	

7.3.3.2 TCI-CH provided

This clause specifies the operations the CH shall provide to the TE.

7.3.3.2.1 tciSendConnected

Signature	void tciSendConnected (in TriPortIdType sender,
	                       in TriComponentIdType receiver,
	                       in Value sendMessage)	

In Parameters	sender	Port identifier at the sending component via which the message is sent.
	receiver	Identifier of the receiving component.
	sendMessage	The message to be sent.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 unicast send operation on a component port, which has been connected to another component port.	
Effect	Sends an asynchronous transmission only to the given receiver component. CH transmits the message to the remote TE on which receiver is being executed and enqueues the data in the remote TE.	

7.3.3.2.2 tciSendConnectedBC

Signature	void tciSendConnectedBC (in TriPortIdType sender, in Value sendMessage)	
In Parameters	sender	Port identifier at the sending component via which the message is sent.
	sendMessage	The message to be sent.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 broadcast send operation on a component port, which has been connected to other component ports.	
Effect	Sends an asynchronous transmission to all components being connected to this port. CH transmits the message to all remote TEs on which receivers are being executed and enqueues the data in the remote TEs.	

7.3.3.2.3 tciSendConnectedMC

Signature	void tciSendConnectedMC (in TriPortIdType sender,
	                         in TriComponentIdListType receivers,
	                         in Value sendMessage)	

In Parameters	sender	Port identifier at the sending component via which the message is sent.
	receivers	Identifiers of the receiving components.
	sendMessage	The message to be sent.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 multicast send operation on a component port, which has been connected to other component ports.	
Effect	Sends an asynchronous transmission to all given receiver components. CH transmits the message to all remote TEs on which receivers are being executed and enqueues the data in the remote TEs.	

7.3.3.2.4 tciCallConnected

Signature	`void tciCallConnected (in TriPortIdType sender,`
	 `                        in TriComponentIdType receiver,`
	 `                        in TriSignatureIdType signature,`
	 `                        in TciParameterListType parameterList)`	

In Parameters	`sender`	Port identifier at the sending component via which the message is sent.
	`receiver`	Identifier of the receiving component.
	`signature`	Identifier of the signature of the procedure call.
	`parameterList`	A list of value parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 unicast call operation on a component port, which has been connected to another component port. All *in* and *inout* procedure parameters contain values. All *out* procedure parameters shall contain the distinct value of `null` because they are only of relevance in a reply to the procedure call but not in the procedure call itself. The procedure parameters are the parameters specified in the TTCN-3 signature template.	
Effect	On invocation of this operation the TE can initiate the procedure call corresponding to the signature identifier `signature` at the called component receiver. The `tciCallConnected` operation shall return without waiting for the return of the issued procedure call. Note that an optional timeout value, which can be specified in the TTCN-3 ATS for a call operation, is not included in the `tciCallConnected` operation signature. The TE is responsible to address this issue by starting a timer for the TTCN-3 call operation in the PA with a separate TRI operation call, i.e., `triStartTimer`. CH transmits the call to the remote TE on which `receiver` is being executed and enqueues the call in the remote TE.	

7.3.3.2.5 tciCallConnectedBC

Signature	`void tciCallConnectedBC (in TriPortIdType sender,`
	 `                          in TriSignatureIdType signature,`
	 `                          in TciParameterListType parameterList)`	

In Parameters	`sender`	Port identifier at the sending component via which the message is sent.
	`signature`	Identifier of the signature of the procedure call.
	`parameterList`	A list of value parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 broadcast call operation on a component port, which has been connected to other component ports. All *in* and *inout* procedure parameters contain values. All *out* procedure parameters shall contain the distinct value of `null` because they are only of relevance in a reply to the procedure call but not in the procedure call itself. The procedure parameters are the parameters specified in the TTCN-3 signature template.	

Effect	On invocation of this operation the TE can initiate the procedure call corresponding to the signature identifier `signature` at the called component receiver. The `tciCallConnected` operation shall return without waiting for the return of the issued procedure call. Note that an optional timeout value, which can be specified in the TTCN-3 ATS for a call operation, is not included in the `tciCallConnected` operation signature. The TE is responsible to address this issue by starting a timer for the TTCN-3 call operation in the PA with a separate TRI operation call, i.e., `triStartTimer`. CH transmits the call to all remote TEs on which a `receiver` is being executed and enqueues the call in the remote TEs.

7.3.3.2.6 tciCallConnectedMC

Signature	`void tciCallConnectedMC (in TriPortIdType sender,`
	`                         in TriComponentIdListType receivers,`
	`                         in TriSignatureIdType signature,`
	`                         in TciParameterListType parameterList)`	

In Parameters	`sender`	Port identifier at the sending component via which the message is sent.
	`receivers`	Identifier of the receiving components.
	`signature`	Identifier of the signature of the procedure call.
	`parameterList`	A list of value parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 multicast call operation on a component port, which has been connected to other component ports. All *in* and *inout* procedure parameters contain values. All *out* procedure parameters shall contain the distinct value of `null` because they are only of relevance in a reply to the procedure call but not in the procedure call itself. The procedure parameters are the parameters specified in the TTCN-3 signature template.	
Effect	On invocation of this operation the TE can initiate the procedure call corresponding to the signature identifier `signature` at the called component receiver. The `tciCallConnected` operation shall return without waiting for the return of the issued procedure call. Note that an optional timeout value, which can be specified in the TTCN-3 ATS for a call operation, is not included in the `tciCallConnected` operation signature. The TE is responsible to address this issue by starting a timer for the TTCN-3 call operation in the PA with a separate TRI operation call, i.e., `triStartTimer`. CH transmits the call to all remote TEs on which a `receiver` is being executed and enqueues the call in the remote TEs.	

7.3.3.2.7 tciReplyConnected

Signature	void tciReplyConnected (in TriPortIdType sender,
	                        in TriComponentIdType receiver,
	                        in TriSignatureIdType signature,
	                        in TciParameterListType parameterList,
	                        in Value returnValue)	

In Parameters	sender	Identifier of the port sending the reply.
	receiver	Identifier of the component receiving the reply.
	signature	Identifier of the signature of the procedure call.
	parameterList	A list of encoded parameters which are part of the indicated signature. The parameters in parameterList are ordered as they appear in the TTCN-3 signature declaration.
	returnValue	(Optional) return value of the procedure call.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 unicast reply operation on a component port which has been connected to another component port. All *out* and *inout* procedure parameters and the return value contain values. All *in* procedure parameters shall contain the distinct value of null since they are only of relevance to the procedure call but not in the reply to the call. The parameterList contains procedure call parameters. These parameters are the parameters specified in the TTCN-3 signature template. If no return type has been defined for the procedure signature in the TTCN-3 ATS, the distinct value null shall be passed for the return value.	
Effect	On invocation of this operation the CH can issue the reply to a procedure call corresponding to the signature identifier signature and component identifier receiver. CH transmits the reply to the remote TE on which receiver is being executed and enqueues the reply in the remote TE.	

7.3.3.2.8 tciReplyConnectedBC

Signature	void tciReplyConnectedBC (in TriPortIdType sender,
	                          in TriSignatureIdType signature,
	                          in TciParameterListType
	                          parameterList,
	                          in Value returnValue)	

In Parameters	sender	Identifier of the port sending the reply.
	signature	Identifier of the signature of the procedure call.
	parameterList	A list of encoded parameters which are part of the indicated signature. The parameters in parameterList are ordered as they appear in the TTCN-3 signature declaration.
	returnValue	(Optional) return value of the procedure call.
Return Value	void	

Constraint	This operation shall be called by the TE when it executes a TTCN-3 broadcast reply operation on a component port which has been connected to other component ports.
	All *out* and *inout* procedure parameters and the return value contain values. All *in* procedure parameters shall contain the distinct value of `null` since they are only of relevance to the procedure call but not in the reply to the call. The `parameterList` contains procedure call parameters. These parameters are the parameters specified in the TTCN-3 signature template. If no return type has been defined for the procedure signature in the TTCN-3 ATS, the distinct value `null` shall be passed for the return value.
Effect	On invocation of this operation the CH can issue the reply to a procedure call corresponding to the signature identifier `signature` and all components connected to `sender`. CH transmits the exception to all remote TEs on which receivers are being executed and enqueues the exception in the remote TEs.

7.3.3.2.9   tciReplyConnectedMC

Signature	`void tciReplyConnectedMC (in TriPortIdType sender,`
	`                          in TriComponentIdListType receivers,`
	`                          in TriSignatureIdType signature,`
	`                          in TciParameterListType parameterList,`
	`                          in Value returnValue)`	

In Parameters	sender	Identifier of the port sending the reply.
	receivers	Identifier of the components receiving the reply.
	signature	Identifier of the signature of the procedure call.
	parameterList	A list of encoded parameters which are part of the indicated signature. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature declaration.
	returnValue	(Optional) return value of the procedure call.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 multicast reply operation on a component port which has been connected to other component ports.	
	All *out* and *inout* procedure parameters and the return value contain values. All *in* procedure parameters shall contain the distinct value of `null` since they are only of relevance to the procedure call but not in the reply to the call. The `parameterList` contains procedure call parameters. These parameters are the parameters specified in the TTCN-3 signature template. If no return type has been defined for the procedure signature in the TTCN-3 ATS, the distinct value `null` shall be passed for the return value.	
Effect	On invocation of this operation the CH can issue the reply to a procedure call corresponding to the signature identifier `signature` and one of the component identifier in `receivers`. CH transmits the reply to the remote TEs on which `receivers` are being executed and enqueues the reply in the remote TEs.	

7.3.3.2.10   tciRaiseConnected

Signature	`void tciRaiseConnected (in TriPortIdType sender,`
	`                         in TriComponentIdType receiver,`
	`                         in TriSignatureIdType signature,`
	`                         in Value exception)`	

In Parameters	`sender`	Identifier of the port sending the reply.
	`receiver`	Identifier of the component receiving the reply.
	`signature`	Identifier of the signature of the procedure call.
	`exception`	The exception value.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 unicast raise operation on a component port which has been connected to another component port.	
Effect	On invocation of this operation the CH can raise an exception to a procedure call corresponding to the signature identifier `signature` and component identifier `receiver`. CH transmits the exception to the remote TE on which `receiver` is being executed and enqueues the exception in the remote TE.	

7.3.3.2.11   tciRaiseConnectedBC

Signature	`void tciRaiseConnectedBC (in TriPortIdType sender,`
	`                           in TriSignatureIdType signature,`
	`                           in Value exception)`	

In Parameters	`sender`	Identifier of the port sending the reply.
	`signature`	Identifier of the signature of the procedure call.
	`exception`	The exception value.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 broadcast raise operation on a component port which has been connected to other component ports.	
Effect	On invocation of this operation the CH can raise an exception to a procedure call corresponding to the signature identifier `signature` and all components connected to `sender`. CH transmits the exception to all remote TEs on which receivers are being executed and enqueues the exception in the remote TEs.	

7.3.3.2.12   tciRaiseConnectedMC

Signature	`void tciRaiseConnectedMC (in TriPortIdType sender,`
	`                           in TriComponentIdListType receivers,`
	`                           in TriSignatureIdType signature,`
	`                           in Value exception)`	

In Parameters	`sender`	Identifier of the port sending the reply.
	`receivers`	Identifiers of the component receiving the reply.
	`signature`	Identifier of the signature of the procedure call.
	`exception`	The exception value.
Return Value	`void`	

Constraint	This operation shall be called by the TE when it executes a TTCN-3 multicast raise operation on a component port which has been connected to another component port.
Effect	On invocation of this operation the CH can raise an exception to a procedure call corresponding to the signature identifier `signature` and one of the component identifier `receivers`. CH transmits the exception to all remote TEs on which `receivers` are being executed and enqueues the exception in the remote TEs.

7.3.3.2.13 tciCreateTestComponentReq

Signature	`TriComponentIdType tciCreateTestComponentReq` `(in TciTestComponentKindType kind,` `in Type componentType,` `in TString name)`	
In Parameters	`kind`	The kind of component that shall be created, either `MTC`, `PTC` or `CONTROL`.
	`componentType`	Identifier of the TTCN-3 component type that shall be created.
Return Value	A `TriComponentIdType` value for the created component.	
Constraint	This operation shall be called from the TE when a component has to be created, either explicitly when the TTCN-3 create operation is called or implicitly when the master test component (MTC) or a control component has to be created. `name` shall be set to the distinct value `null` if no name is given in the TTCN-3 create statement.	
Effect	CH transmits the component creation request to the remote TE and calls there the `tciCreateTestComponent` operation to obtain a component identifier for this component.	

7.3.3.2.14 tciStartTestComponentReq

Signature	`void tciStartTestComponentReq(in TriComponentIdType component,` `in TciBehaviourIdType behaviour,` `in TciParamaterListType` `parameterList)`	
In Parameters	`component`	Identifier of the component to be started.
	`behaviour`	Identifier of the behaviour to be started on the component.
	`parameterList`	A list of `Values` where each value defines a parameter from the parameter list as defined in the TTCN-3 function declaration of the function being started. The parameters in `parameterList` are ordered as they appear in the TTCN-3 signature of the test case. If no parameters have to be passed either the `null` value or an empty `parameterList`, i.e., a list of length zero shall be passed.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes the TTCN-3 start operation. Since only *in* parameters are allowed for functions being started ([ITU-T Z.161]), `parameterList` contains only *in* parameters.	
Effect	CH transmits the start component request to the remote TE and calls there the `tciStartTestComponent` operation.	

7.3.3.2.15 tciStopTestComponentReq

Signature	`void tciStopTestComponentReq(in TriComponentIdType component)`	
In Parameters	`component`	Identifier of the component to be stopped.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes the TTCN-3 stop operation.	
Effect	CH transmits the stop component request to the remote TE and calls there the `tciStopTestComponent` operation.	

7.3.3.2.16 tciConnectReq

Signature	`void tciConnectReq (in TriPortIdType fromPort,` `                    in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be connected from.
	`toPort`	Identifier of the test component port to be connected to.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 connect operation.	
Effect	CH transmits the connection request to the remote TE where it calls the `tciConnect` operation to establish a logical connection between the two indicated ports. Note that both ports can be on remote TEs. In this case, the operation returns only after calling the `tciConnect` operation on both remote TEs.	

7.3.3.2.17 tciDisconnectReq

Signature	`void tciDisconnectReq (in TriPortIdType fromPort,` `                        in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be disconnected.
	`toPort`	Identifier of the test component port to be disconnected.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 disconnect operation.	
Effect	CH transmits the disconnect request to the remote TE where it calls the `tciDisconnect` operation to tear down the logical connection between the two indicated ports. Note that both ports can be on remote TEs. In this case, the operation returns only after calling the `tciDisconnect` operation on both remote TEs.	

7.3.3.2.18 tciMapReq

Signature	`void tciMapReq (in TriPortIdType fromPort,` `                in TriPortIdType toPort)`	
In Parameters	`fromPort`	Identifier of the test component port to be mapped from.
	`toPort`	Identifier of the test component port to be mapped to.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 map operation.	
Effect	CH transmits the map request to the remote TE where it calls the `tciMap` operation to establish a logical connection between the two indicated ports.	

7.3.3.2.19 tciUnmapReq

Signature	void tciUnmapReq (in TriPortIdType fromPort, in TriPortIdType toPort)	
In Parameters	`fromPort`	Identifier of the test component port to be unmapped.
	`toPort`	Identifier of the test component port to be unmapped.
Return Value	`void`	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 unmap operation.	
Effect	CH transmits the unmap request to the remote TE where it calls the `tciUnmap` operation to tear down the logical connection between the two indicated ports.	

7.3.3.2.20 tciTestComponentTerminatedReq

Signature	void tciTestComponentTerminatedReq (in TriComponentIdType component, in VerdictValue verdict)	
In Parameters	`component`	Identifier of the component that has terminated.
	`verdict`	Verdict after termination of the component.
Return Value	`void`	
Constraint	This operation shall be called by the TE when a test component terminates execution, either explicitly with the TTNC-3 stop operation or implicitly, if it has reached the last statement.	
Effect	The CH is notified of the termination of the indicated test component. Since a function being executed on a test component can only have *in* parameters ([ITU-T Z.161]), the `tciTestComponentTerminateReq` operation does not have a `parameterList` parameter. CH communicates the termination of the indicated component to all participating TEs and to the special TE*, which keeps track of the overall verdict.	

7.3.3.2.21 tciTestComponentRunningReq

Signature	TBoolean tciTestComponentRunningReq (in TriComponentIdType component)	
In Parameters	`component`	Identifier of the component to be checked for running.
Return Value	`true` if the indicated component is still executing a behaviour, `false` otherwise.	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 running operation.	
Effect	CH transmits the running request to the remote TE having the test component to be checked, where it calls the `tciTestComponentRunning` operation to check the execution status of the indicated test component.	

7.3.3.2.22 tciTestComponentDoneReq

Signature	TBoolean tciTestComponentDoneReq (in TriComponentIdType comp)	
In Parameters	comp	Identifier of the component to be checked for done.
Return Value	true if the indicated component has completed executing its behaviour, false otherwise.	
Constraint	This operation shall be called by the TE when it executes a TTCN-3 done operation.	
Effect	CH transmits the done request to the remote TE having the test component to be checked, where it calls the tciTestComponentDone operation to check the status of the indicated test component.	

7.3.3.2.23 tciGetMTCReq

Signature	TriComponentIdType tciGetMTCReq()
Return Value	A TriComponentIdType value of the MTC.
Constraint	This operation shall be called by the TE when it executes a TTCN-3 mtc operation.
Effect	The CH determines the component id of the MTC.

7.3.3.2.24 tciExecuteTestCaseReq

Signature	void tciExecuteTestCaseReq (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)	
In Parameters	testCaseId	A test case identifier as defined in the TTCN-3 module.
	tsiPortList	tsiPortList contains all ports that have been declared in the definition of the system component for the test case, i.e., the TSI ports. If a system component has not been explicitly defined for the test case, then the tsiPortList contains all communication ports of the MTC. The ports in tsiPortList are ordered as they appear in the respective TTCN-3 component type declaration. If no ports have to be passed either the null value or an empty tsiPortList, i.e., a list of length zero shall be passed.
Return Value	void	
Constraint	This operation can be called by the TE immediately before it starts the test case behaviour on the MTC (in course of a TTCN-3 execute operation).	
Effect	CH transmits the execute test case request to the remote TEs having system ports of the indicated test case. Static connections to the SUT and the initialization of communication means for TSI ports can be set up.	

7.3.3.2.25 tciResetReq

Signature	void tciResetReq ()
Return Value	void
Constraint	This operation can be called by the TE at any time to reset the test system.
Effect	CH transmits the reset request to all involved TEs.

7.3.3.2.26 tciKillTestComponentReq

Signature	void tciKillTestComponentReq(in TriComponentIdType comp)	
In Parameters	comp	Identifier of the component to be killed.
Return Value	void	
Constraint	This operation shall be called by the TE when it executes the TTCN-3 kill operation.	
Effect	CH transmits the kill component request to the remote TE and calls there the tciKillTestComponent operation.	

7.3.3.2.27 tciTestComponentAliveReq

Signature	TBoolean tciTestComponentAliveReq (in TriComponentIdType comp)	
In Parameters	comp	Identifier of the component to be checked for being alive.
Return Value	true if the indicated component is alive, false otherwise.	
Constraint	This operation shall be called by the TE when it executes the TTCN-3 alive operation.	
Effect	CH transmits the request to the remote TE that created the test component in question, where it calls the tciTestComponentAlive operation to check the status of the indicated test component.	

7.3.3.2.28 tciTestComponentKilledReq

Signature	TBoolean tciTestComponentKilledReq (in TriComponentIdType comp)	
In Parameters	comp	Identifier of the component to be checked for being killed.
Return Value	true if the indicated component has been killed, false otherwise.	
Constraint	This operation shall be called by the TE when it executes the TTCN-3 killed operation.	
Effect	CH transmits the request to the remote TE that created the test component in question, where it calls the tciTestComponentKilled operation to check the status of the indicated test component.	

7.3.4 The TCI-TL interface

The TCI test logging interface (TCI-TL) describes the operations a TTCN-3 Executable is required to implement and the operations a test logging implementation shall provide to the TE (Figure 8).
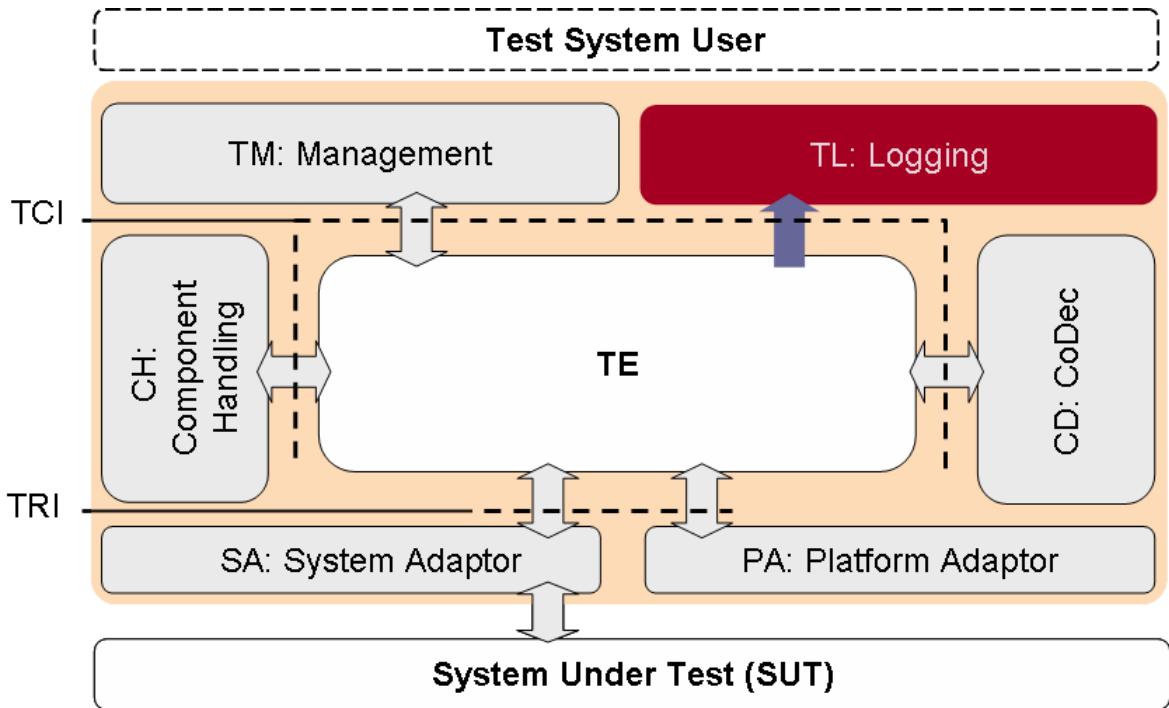
Figure 8 – The TCI-TL interface

The logging provides for all TTCN-3 level operations an operation to log the respective event being performed by the TE, the SA, the PA, the CH or the CD to the user.

7.3.4.1 TCI-TL provided

This clause specifies the operations the TL shall provide to the TE.

7.3.4.1.1 tliTcExecute

Signature	`void tliTcExecute(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line, in`
	`                   TriComponentIdType c,`
	`                   in TciTestCaseIdType tcId, in`
	`                   TriParameterListType triPars,`
	`                   in TriTimerDurationType dur)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`tcId`	The testcase to be executed.
	`triPars`	The list of parameters required by the testcase.
	`dur`	Duration of the execution.
Return Value	`void`	
Constraint	Shall be called by TE to log the execute test case request.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.2 tliTcStart

Signature	void tliTcStart(in TString am, in TInteger ts,
	                in TString src, in TInteger line, in
	                TriComponentIdType c,
	                in TciTestCaseIdType tcId, in
	                TciParameterListType tciPars,
	                in TriTimerDurationType dur)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	tcId	The testcase to be executed.
	tciPars	The list of parameters required by the testcase.
	dur	Duration of the execution.
Return Value	void	
Constraint	Shall be called by TE to log the start of a testcase. This event occurs before the testcase is started.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.3 tliTcStop

Signature	void tliTcStop(in TString am, in TInteger ts,
	               in TString src, in TInteger line,
	               in TriComponentIdType c)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log the stop of a testcase.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.4 tliTcStarted

Signature	`void tliTcStarted(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c, in`
	`                   TciTestCaseIdType tcId,`
	`                   in TciParameterListType tciPars, in`
	`                   TriTimerDurationType dur)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`tcId`	The testcase to be executed.
	`tciPars`	The list of parameters required by the testcase.
	`dur`	Duration of the execution.
Return Value	`Void`	
Constraint	Shall be called by TM to log the start of a testcase. This event occurs after the testcase was started.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.5 tliTcTerminated

Signature	`void tliTcTerminated(in TString am, in TInteger ts,`
	`                      in TString src, in TInteger line,`
	`                      in TriComponentIdType c,`
	`                      in TciTestCaseIdType tcId,`
	`                      in TciParameterListType tciPars,`
	`                      in VerdictValue verdict)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`tcId`	The testcase to be executed.
	`tciPars`	The list of parameters required by the testcase.
	`verdict`	The verdict of the testcase.
Return Value	`void`	
Constraint	Shall be called by TM to log the termination of a testcase. This event occurs after the testcase terminated.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.6   tliCtrlStart

Signature	void tliCtrlStart(in TString am, in TInteger ts,
	                   in TString src, in TInteger line,
	                   in TriComponentIdType c)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log the start of the control part. This event occurs before the control is started. If the control is not represented by a TRI component, c is null.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.7   tliCtrlStop

Signature	void tliCtrlStop(in TString am, in TInteger ts,
	                  in TString src, in TInteger line,
	                  in TriComponentIdType c)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log the stop of the control part. This event occurs before the control is stopped. If the control is not represented by a TRI component, c is null.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.8 tliCtrlTerminated

Signature	void tliCtrlTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TM to log the termination of the control part. This event occurs after the control has terminated. If the control is not represented by a TRI component, c is null.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.9 tliMSend_m

Signature	void tliMSend_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue, in TriAddressType address, in TciStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	to	The port to which the message is sent.
	msgValue	The value to be encoded and sent.
	address	The address of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded message.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a unicast send operation. This event occurs after sending. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.10 tliMSend_m_BC

Signature	`void tliMSend_m_BC(in TString am, in TInteger ts,`
	`                    in TString src, in TInteger line, in`
	`                    TriComponentIdType c,`
	`                    in TriPortIdType at, in TriPortIdType to,`
	`                    in Value msgValue,`
	`                    in TciStatusType encoderFailure,`
	`                    in TriMessageType msg,`
	`                    in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the message is sent.
	`to`	The port to which the message is sent.
	`msgValue`	The value to be encoded and sent.
	`encoderFailure`	The failure message which might occur at encoding.
	`msg`	The encoded message.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by SA to log a broadcast send operation. This event occurs after sending. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.11 tliMSend_m_MC

Signature	`void tliMSend_m_MC(in TString am, in TInteger ts,`
	`                    in TString src, in TInteger line, in`
	`                    TriComponentIdType c,`
	`                    in TriPortIdType at, in TriPortIdType to,`
	`                    in Value msgValue,`
	`                    in TriAddressListType addresses,`
	`                    in TciStatusType encoderFailure,`
	`                    in TriMessageType msg,`
	`                    in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	to	The port to which the message is sent.
	msgValue	The value to be encoded and sent.
	addresses	The addresses of the destinations within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded message.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a multicast send operation. This event occurs after sending. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.12 tliMSend_c

Signature	`void tliMSend_c(in TString am, in TInteger ts,`
	`                 in TString src, in TInteger line,`
	`                 in TriComponentIdType c,`
	`                 in TriPortIdType at, in TriPortIdType to,`
	`                 in Value msgValue,`
	`                 in TriStatusType transmissionFailure)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	msgValue	The value to be encoded and sent.
	to	The component which will receive the message.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a unicast send operation. This event occurs after sending. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.13 tliMSend_c_BC

Signature	`void tliMSend_c_BC(in TString am, in TInteger ts,`
	`in TString src, in TInteger line,`
	`in TriComponentIdType c,`
	`in TriPortIdType at, in TriPortIdListType to,`
	`in Value msgValue,`
	`in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the message is sent.
	`to`	The ports to which the message is sent.
	`msgValue`	The value to be encoded and sent.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by CH to log a broadcast send operation. This event occurs after sending. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.14 tliMSend_c_MC

Signature	`void tliMSend_c_MC(in TString am, in TInteger ts,`
	`in TString src, in TInteger line,`
	`in TriComponentIdType c,`
	`in TriPortIdType at,`
	`in TriPortIdListType to,`
	`in Value msgValue,`
	`in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the message is sent.
	`to`	The port to which the message is sent.
	`msgValue`	The value to be encoded and sent.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by CH to log a multicast send operation. This event occurs after sending. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.15 tliMDetected_m

Signature	void tliMDetected_m(in TString am, in TInteger ts,
	                    in TString src, in TInteger line,
	                    in TriComponentIdType c,
	                    in TriPortIdType at,
	                    in TriPortIdType from,
	                    in TriMessageType msg,
	                    in TriAddressType address)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	from	The port from which the message has been sent.
	msg	The received encoded message.
	address	The address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by SA to log the enqueuing of a message. This event occurs after the message is enqueued. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.16 tliMDetected_c

Signature	void tliMDetected_c(in TString am, in TInteger ts,
	                    in TString src, in TInteger line,
	                    in TriComponentIdType c,
	                    in TriPortIdType at,
	                    in TriPortIdType from, in Value msgValue)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	from	The port from which the message has been sent.
	msgValue	The received message.
Return Value	void	
Constraint	Shall be called by CH to log the enqueuing of a message. This event occurs after the message is enqueued. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.17 tliMMismatch_m

Signature	`void tliMMismatch_m(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at, in Value msgValue,`
	`                     in TciValueTemplate msgTmpl,`
	`                     in TciValueDifferenceList diffs,`
	`                     in TriAddressType address,`
	`                     in TciValueTemplate addressTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the message is received.
	`msgValue`	The message which is checked against the template.
	`msgTmpl`	The template used to check the message match.
	`diffs`	The difference/the mismatch between message and template
	`address`	The address of the source within the SUT.
	`addressTmpl`	The expected address of the source within the SUT.
Return Value	`void`	
Constraint	Shall be called by TE to log the mismatch of a template. This event occurs after checking a template match. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.18 tliMMismatch_c

Signature	`void tliMMismatch_c(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at, in Value msgValue,`
	`                     in TciValueTemplate msgTmpl,`
	`                     in TciValueDifferenceList diffs,`
	`                     in TriComponentIdType from,`
	`                     in TciNonValueTemplate fromTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the message is received.
	`msgValue`	The message which is checked against the template.
	`msgTmpl`	The template used to check the message match.
	`diffs`	The difference/the mismatch between message and template
	`from`	The component which sent the message.
	`fromTmpl`	The expected sender component.

Return Value	void
Constraint	Shall be called by TE to log the mismatch of a template. This event occurs after checking a template match. This event is used for logging the intercomponent communication.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.19 tliMReceive_m

Signature	void tliMReceive_m(in TString am, in TInteger ts,
	                    in TString src, in TInteger line,
	                    in TriComponentIdType c,
	                    in TriPortIdType at,
	                    in Value msgValue,
	                    in TciValueTemplate msgTmpl,
	                    in TriAddressType address,
	                    in TciValueTemplate addressTmpl)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	msgValue	The message which is checked against the template.
	msgTmpl	The template used to check the message match.
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by TE to log the receipt of a message. This event occurs after checking a template match. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.20 tliMReceive_c

Signature	void tliMReceive_c(in TString am, in TInteger ts,
	                    in TString src, in TInteger line,
	                    in TriComponentIdType c,
	                    in TriPortIdType at,
	                    in Value msgValue,
	                    in TciValueTemplate msgTmpl,
	                    in TriComponentIdType fromComp,
	                    in TciNonValueTemplate fromTmpl)

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	msg	The message which is checked against the template.
	msgTmpl	The template used to check the message match.
	fromComp	The component which sent the message.
	fromTmpl	The expected sender component.
Return Value	void	
Constraint	Shall be called by TE to log the receipt of a message. This event occurs after checking a template match. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.21   tliPrCall_m

Signature	void tliPrCall_m(in TString am, in TInteger ts,
	                  in TString src, in TInteger line,
	                  in TriComponentIdType c,
	                  in TriPortIdType at, in TriPortIdType to,
	                  in TriSignatureIdType signature,
	                  in TciParameterListType tciPars,
	                  in TriAddressType address,
	                  in TciStatusType encoderFailure,
	                  in TriParameterListType triPars,
	                  in TriStatusType transmissionFailure)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	address	The address of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	transmissionFailure	The failure message which might occur at transmission.

Return Value	void
Constraint	Shall be called by SA to log a unicast call operation. This event occurs after call execution. This event is used for logging the communication with the SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.22 tliPrCall_m_BC

Signature	`void tliPrCall_m_BC(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at, in TriPortIdType to,`
	`                     in TriSignatureIdType signature,`
	`                     in TciParameterListType tciPars,`
	`                     in TciStatusType encoderFailure,`
	`                     in TriParameterListType triPars,`
	`                     in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	transmissionFailure	The failure message which might occur at transmission.

Return Value	void
Constraint	Shall be called by SA to log a broadcast call operation. This event occurs after call execution. This event is used for logging the communication with the SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.23 tliPrCall_m_MC

Signature	`void tliPrCall_m_MC(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at, in TriPortIdType to,`
	`                     in TriSignatureIdType signature,`
	`                     in TciParameterListType tciPars,`
	`                     in TriAddressListType addresses,`
	`                     in TciStatusType encoderFailure,`
	`                     in TriParameterListType triPars,`
	`                     in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	addresses	The addresses of the destinations within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a multicast call operation. This event occurs after call execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.24   tliPrCall_c

Signature	void tliPrCall_c(in TString am, in TInteger ts, in TString src,
	                  in TInteger line, in TriComponentIdType c,
	                  in TriPortIdType at, in TriPortIdType to,
	                  in TriSignatureIdType signature,
	                  in TciParameterListType tciPars,
	                  in TriStatusType transmissionFailure)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a unicast call operation. This event occurs after call execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.25 tliPrCall_c_BC

Signature	void tliPrCall_c_BC(in TString am, in TInteger ts, in TString src,
	                     in TInteger line, in TriComponentIdType c,
	                     in TriPortIdType at, in TriPortIdListType to,
	                     in TriSignatureIdType signature,
	                     in TciParameterListType tciPars,
	                     in TriStatusType transmissionFailure)	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the call is invoked.
	`to`	The port list to which the call is sent.
	`signature`	The signature of the called operation.
	`tciPars`	The parameters of the called operation.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`Void`	
Constraint	Shall be called by CH to log a broadcast call operation. This event occurs after call execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.26 tliPrCall_c_MC

Signature	void tliPrCall_c_MC(in TString am, in TInteger ts,
	                     in TString src, in TInteger line,
	                     in TriComponentIdType c,
	                     in TriPortIdType at,
	                     in TriPortIdListType to,
	                     in TriSignatureIdType signature,
	                     in TciParameterListType tciPars,
	                     in TriStatusType transmissionFailure)	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the call is invoked.
	`to`	The port list to which the call is sent.
	`signature`	The signature of the called operation.
	`tciPars`	The parameters of the called operation.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by CH to log a multicast call operation. This event occurs after call execution. This event is used for logging the intercomponent communication.	

Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.27 tliPrGetCallDetected_m

Signature	`void tliPrGetCallDetected_m(in TString am, in TInteger ts,`
	`                             in TString src,`
	`                             in TInteger line,`
	`                             in TriComponentIdType c,`
	`                             in TriPortIdType at,`
	`                             in TriPortIdType from,`
	`                             in TriSignatureIdType signature,`
	`                             in TriParameterListType triPars,`
	`                             in TriAddressType address)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	from	The port from which the call has been sent.
	signature	The signature of the detected call.
	triPars	The encoded parameters of detected call.
	address	The address of the destination within the SUT.
Return Value	`void`	
Constraint	Shall be called by SA to log the getcall enqueue operation. This event occurs after call is enqueued. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.28 tliPrGetCallDetected_c

Signature	`void tliPrGetCallDetected_c(in TString am, in TInteger ts,`
	`                             in TString src, in TInteger line,`
	`                             in TriComponentIdType c,`
	`                             in TriPortIdType at,`
	`                             in TriPortIdType from,`
	`                             in TriSignatureIdType signature,`
	`                             in TciParameterListType tciPars)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	from	The port from which the call has been sent.
	signature	The signature of the called operation.
	tciPars	The encoded parameters of the detected call.
Return Value	`void`	

Constraint	Shall be called by CH to log the getcall enqueue operation. This event occurs after call is enqueued. This event is used for logging the intercomponent communication.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.29 tliPrGetCallMismatch_m

Signature	`void tliPrGetCallMismatch_m(in TString am, in TInteger ts,`
	`                            in TString src,`
	`                            in TInteger line,`
	`                            in TriComponentIdType c,`
	`                            in TriPortIdType at,`
	`                            in TriSignatureIdType signature,`
	`                            in TciParameterListType tciPars,`
	`                            in TciValueTemplate parsTmpl,`
	`                            in TciValueDifferenceList diffs,`
	`                            in TriAddressType address,`
	`                            in TciValueTemplate addressTmpl)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	signature	The signature of the detected call.
	tciPars	The parameters of the detected call.
	parsTmpl	The template used to check the parameter match.
	diffs	The difference/the mismatch between call and template
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	`void`	
Constraint	Shall be called by TE to log the mismatch of a getcall. This event occurs after getcall is checked against a template. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.30 tliPrGetCallMismatch_c

Signature	`void tliPrGetCallMismatch_c(in TString am, in TInteger ts,`
	`                            in TString src,`
	`                            in TInteger line,`
	`                            in TriComponentIdType c,`
	`                            in TriPortIdType at,`
	`                            in TriSignatureIdType signature,`
	`                            in TciParameterListType tciPars,`
	`                            in TciValueTemplate parsTmpl,`
	`                            in TciValueDifferenceList diffs,`
	`                            in TriComponentIdType from,`
	`                            in TciNonValueTemplate fromTmpl)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	signature	The signature of the detected call.
	tciPars	The parameters of the detected call.
	parsTmpl	The template used to check the parameter match.
	diffs	The difference/the mismatch between message and template
	from	The component which called the operation.
	fromTmpl	The expected calling component.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a getcall. This event occurs after getcall is checked against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.31 tliPrGetCall_m

Signature	```void tliPrGetCall_m(in TString am, in TInteger ts,```
	```in TString src, in TInteger line,```
	```in TriComponentIdType c,```
	```in TriPortIdType at,```
	```in TriSignatureIdType signature,```
	```in TciParameterListType tciPars,```
	```in TciValueTemplate parsTmpl,```
	```in TriAddressType address,```
	```in TciValueTemplate addressTmpl)```	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	signature	The signature of the detected call.
	tciPars	The parameters of the detected call.
	parsTmpl	The template used to check the parameter match.
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by TE to log getting a call. This event occurs after getcall has matched against a template. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.32 tliPrGetCall_c

Signature	`void tliPrGetCall_c(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at,`
	`                     in TriSignatureIdType signature,`
	`                     in TciParameterListType tciPars,`
	`                     in TciValueTemplate parsTmpl,`
	`                     in TriComponentIdType from,`
	`                     in TciNonValueTemplate fromTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the call is received.
	`signature`	The signature of the detected call.
	`tciPars`	The parameters of the detected call.
	`parsTmpl`	The template used to check the parameter match.
	`from`	The component which called the operation.
	`fromTmpl`	The expected calling component.
Return Value	`void`	
Constraint	Shall be called by TE to log getting a call. This event occurs after getcall has matched against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.33 tliPrReply_m

Signature	`void tliPrReply_m(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c,`
	`                   in TriPortIdType at, in TriPortIdType to,`
	`                   in TriSignatureIdType signature,`
	`                   in TciParameterListType tciPars,`
	`                   in Value replValue,`
	`                   in TriAddressType address,`
	`                   in TciStatusType encoderFailure,`
	`                   in TriParameterListType triPars,`
	`                   in TriParameterType repl,`
	`                   in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the reply is sent.

In Parameters	to	The port to which the reply is sent.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	replValue	The reply to be sent.
	address	The address of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	repl	The encoded reply.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a unicast reply operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.34   tliPrReply_m_BC

| **Signature** | ```
void tliPrReply_m_BC(in TString am, in TInteger ts,
                     in TString src, in TInteger line,
                     in TriComponentIdType c, in TriPortIdType at,
                     in TriPortIdType to,
                     in TriSignatureIdType signature,
                     in TciParameterListType tciPars,
                     in Value replValue,
                     in TciStatusType encoderFailure,
                     in TriParameterListType triPars,
                     in TriParameterType repl,
                     in TriStatusType transmissionFailure)
``` | |
|---|---|---|
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is sent.
	to	The port to which the reply is sent.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	replValue	The reply to be sent.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	repl	The encoded reply.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a broadcast reply operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.35 tliPrReply_m_MC

Signature	`void tliPrReply_m_MC(in TString am, in TInteger ts,`
	`                      in TString src, in TInteger line,`
	`                      in TriComponentIdType c,`
	`                      in TriPortIdType at, in TriPortIdType to,`
	`                      in TriSignatureIdType signature,`
	`                      in TciParameterListType tciPars,`
	`                      in Value replValue,`
	`                      in TriAddressListType addresses,`
	`                      in TciStatusType encoderFailure,`
	`                      in TriParameterListType triPars,`
	`                      in TriParameterType repl,`
	`                      in TriStatusType transmissionFailure)`

In Parameters	`am` — An additional message.
	`ts` — The time when the event is produced.
	`src` — The source file of the test specification.
	`line` — The line number where the request is performed.
	`c` — The component which produces this event.
	`at` — The port via which the reply is sent.
	`to` — The port to which the reply is sent.
	`signature` — The signature relating to the reply.
	`tciPars` — The signature parameters relating to the reply.
	`replValue` — The reply to be sent.
	`addresses` — The addresses of the destinations within the SUT.
	`encoderFailure` — The failure message which might occur at encoding.
	`triPars` — The encoded parameters.
	`repl` — The encoded reply.
	`transmissionFailure` — The failure message which might occur at transmission.
Return Value	`void`
Constraint	Shall be called by SA to log a multicast reply operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.36 tliPrReply_c

Signature	`void tliPrReply_c(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c,`
	`                   in TriPortIdType at, in TriPortIdType to,`
	`                   in TriSignatureIdType signature,`
	`                   in Value parsValue, in Value replValue,`
	`                   in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is sent.
	to	The port to which the reply is sent.
	signature	The signature relating to the reply.
	parsValue	The signature parameters relating to the reply.
	replValue	The reply to be sent.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a unicast reply operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.37 tliPrReply_c_BC

Signature	void tliPrReply_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is sent.
	to	The port list to which the reply is sent.
	signature	The signature relating to the reply.
	parsValue	The signature parameters relating to the reply.
	replValue	The reply to be sent.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a broadcast reply operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.38 tliPrReply_c_MC

Signature	`void tliPrReply_c_MC(in TString am, in TInteger ts,`
	`                     in TString src, in TInteger line,`
	`                     in TriComponentIdType c,`
	`                     in TriPortIdType at,`
	`                     in TriPortIdListType to,`
	`                     in TriSignatureIdType signature,`
	`                     in Value parsValue, in Value replValue,`
	`                     in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the reply is sent.
	`to`	The port list to which the reply is sent.
	`signature`	The signature relating to the reply.
	`parsValue`	The signature parameters relating to the reply.
	`replValue`	The reply to be sent.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by CH to log a multicast reply operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.39 tliPrGetReplyDetected_m

Signature	`void tliPrGetReplyDetected_m(in TString am, in TInteger ts,`
	`                             in TString src, in TInteger line,`
	`                             in TriComponentIdType c,`
	`                             in TriPortIdType at,`
	`                             in TriPortIdType from,`
	`                             in TriSignatureIdType signature,`
	`                             in TriParameterListType triPars,`
	`                             in TriParameterType repl,`
	`                             in TriAddressType address)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the reply is received.
	`from`	The port from which the reply has been sent.
	`signature`	The signature relating to the reply.
	`triPars`	The encoded parameters of detected reply.

In Parameters	repl	The received encoded reply.
	address	The address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by SA to log the getreply enqueue operation. This event occurs after getreply is enqueued. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.40   tliPrGetReplyDetected_c

Signature	```void tliPrGetReplyDetected_c(in TString am, in TInteger ts,``` ```in TString src, in TInteger line,``` ```in TriComponentIdType c,``` ```in TriPortIdType at,``` ```in TriPortIdType from,``` ```in TriSignatureIdType signature,``` ```in TciParameterListType tciPars,``` ```in Value replValue)```	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is received.
	from	The port from which the reply has been sent.
	signature	The signature relating to the reply.
	tciPars	The encoded parameters of detected reply.
	replValue	The received reply.
Return Value	void	
Constraint	Shall be called by CH to log the getreply enqueue operation. This event occurs after getreply is enqueued. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.41   tliPrGetReplyMismatch_m

Signature	```void tliPrGetReplyMismatch_m(in TString am, in TInteger ts,``` ```in TString src, in TInteger line,``` ```in TriComponentIdType c,``` ```in TriPortIdType at,``` ```in TriSignatureIdType signature,``` ```in TciParameterListType tciPars,``` ```in TciValueTemplate parsTmpl,``` ```in Value replValue,``` ```in TciValueTemplate replyTmpl,``` ```in TciValueDifferenceList diffs,``` ```in TriAddressType address,``` ```in TciValueTemplate addressTmpl)```

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is received.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	parsTmpl	The template used to check the parameter match.
	replValue	The received reply.
	replyTmpl	The template used to check the reply match.
	diffs	The difference/the mismatch between reply and template.
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a getreply operation. This event occurs after getreply is checked against a template. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.42 tliPrGetReplyMismatch_c

| **Signature** | ```void tliPrGetReplyMismatch_c(in TString am, in TInteger ts,
                                in TString src, in TInteger line,
                                in TriComponentIdType c,
                                in TriPortIdType at,
                                in TriSignatureIdType signature,
                                in TciParameterListType tciPars,
                                in TciValueTemplate parsTmpl,
                                in Value replValue,
                                in TciValueTemplate replyTmpl,
                                in TciValueDifferenceList diffs,
                                in TriComponentIdType from,
                                in TciNonValueTemplate fromTmpl)``` |
|---|---|

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is received.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	parsTmpl	The template used to check the parameter match.
	replValue	The received reply.
	replyTmpl	The template used to check the reply match.
	diffs	The difference/the mismatch between reply and template.
	from	The component which sent the reply.
	fromTmpl	The expected replying component.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a getreply operation. This event occurs after getreply is checked against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.43 tliPrGetReply_m

| Signature | ```
void tliPrGetReply_m(in TString am, in TInteger ts,
                     in TString src,
                     in TInteger line, in TriComponentIdType c,
                     in TriPortIdType at,
                     in TriSignatureIdType signature,
                     in TciParameterListType tciPars,
                     in TciValueTemplate parsTmpl,
                     in Value replValue,
                     in TciValueTemplate replyTmpl,
                     in TriAddressType address,
                     in TciValueTemplate addressTmpl)
``` | |
|---|---|---|
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is received.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	parsTmpl	The template used to check the parameter match.
	replValue	The received reply.
	replyTmpl	The template used to check the reply match.
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.

Return Value	`void`
Constraint	Shall be called by TE to log getting a reply. This event occurs after getreply is checked against a template. This event is used for logging the communication with SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.44 tliPrGetReply_c

Signature	`void tliPrGetReply_c(in TString am, in TInteger ts,`
	`                      in TString src, in TInteger line,`
	`                      in TriComponentIdType c,`
	`                      in TriPortIdType at,`
	`                      in TriSignatureIdType signature,`
	`                      in TciParameterListType tciPars,`
	`                      in TciValueTemplate parsTmpl,`
	`                      in Value replValue,`
	`                      in TciValueTemplate replyTmpl,`
	`                      in TriComponentIdType from,`
	`                      in TciNonValueTemplate fromTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the reply is received.
	`signature`	The signature relating to the reply.
	`tciPars`	The signature parameters relating to the reply.
	`parsTmpl`	The template used to check the parameter match.
	`replValue`	The received reply.
	`replyTmpl`	The template used to check the reply match.
	`from`	The component which sent the reply.
	`fromTmpl`	The expected replying component.
Return Value	`void`	
Constraint	Shall be called by TE to log getting a reply. This event occurs after getreply is checked against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.45 tliPrRaise_m

Signature	`void tliPrRaise_m(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c,`
	`                   in TriPortIdType at, in TriPortIdType to,`
	`                   in TriSignatureIdType signature,`
	`                   in TciParameterListType tciPars,`
	`                   in Value excValue,`
	`                   in TriAddressType address,`
	`                   in TciStatusType encoderFailure,`
	`                   in TriExceptionType exc,`
	`                   in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is sent.
	to	The port to which the exception is sent.
	signature	The signature relating to the exception.
	tciPars	The signature parameters relating to the exception.
	excValue	The exception to be sent.
	address	The address of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	exc	The encoded exception.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA to log a unicast raise operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.46 tliPrRaise_m_BC

Signature	void tliPrRaise_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value excValue, in TciStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is sent.
	to	The port to which the exception is sent.
	signature	The signature relating to the exception.
	tciPars	The signature parameters relating to the exception.
	excValue	The exception to be sent.
	encoderFailure	The failure message which might occur at encoding.
	exc	The encoded exception.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	

Constraint	Shall be called by SA to log a broadcast raise operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.47 tliPrRaise_m_MC

Signature	`void tliPrRaise_m_MC(in TString am, in TInteger ts,`
	`                      in TString src, in TInteger line,`
	`                      in TriComponentIdType c,`
	`                      in TriPortIdType at, in TriPortIdType to,`
	`                      in TriSignatureIdType signature,`
	`                      in TciParameterListType tciPars,`
	`                      in Value excValue,`
	`                      in TriAddressListType addresses,`
	`                      in TciStatusType encoderFailure,`
	`                      in TriExceptionType exc,`
	`                      in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is sent.
	`to`	The port to which the exception is sent.
	`signature`	The signature relating to the exception.
	`tciPars`	The signature parameters relating to the exception.
	`excValue`	The exception to be sent.
	`addresses`	The addresses of the destinations within the SUT.
	`encoderFailure`	The failure message which might occur at encoding.
	`exc`	The encoded exception.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by SA to log a multicast raise operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.48 tliPrRaise_c

Signature	`void tliPrRaise_c(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c,`
	`                   in TriPortIdType at, in TriPortIdType to,`
	`                   in TriSignatureIdType signature,`
	`                   in TciParameterListType tciPars,`
	`                   in Value excValue,`
	`                   in TriStatusType transmissionFailure)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is sent.
	to	The port to which the exception is sent.
	signature	The signature relating to the exception.
	tciPars	The signature parameters relating to the exception.
	excValue	The exception to be sent.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a unicast raise operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.49   tliPrRaise_c_BC

Signature	void tliPrRaise_c_BC(in TString am, in TInteger ts,
	                     in TString src, in TInteger line,
	                     in TriComponentIdType c,
	                     in TriPortIdType at,
	                     in TriPortIdListType to,
	                     in TriSignatureIdType signature,
	                     in TciParameterListType tciPars,
	                     in Value excValue,
	                     in TriStatusType transmissionFailure)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is sent.
	to	The port list to which the exception is sent.
	signature	The signature relating to the exception.
	tciPars	The signature parameters relating to the exception.
	excValue	The exception to be sent.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by CH to log a broadcast raise operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.50 tliPrRaise_c_MC

Signature	`void tliPrRaise_c_MC(in TString am, in TInteger ts,`
	`                      in TString src, in TInteger line,`
	`                      in TriComponentIdType c,`
	`                      in TriPortIdType at,`
	`                      in TriPortIdListType to,`
	`                      in TriSignatureIdType signature,`
	`                      in TciParameterListType tciPars,`
	`                      in Value excValue,`
	`                      in TriStatusType transmissionFailure)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is sent.
	`to`	The port list to which the exception is sent.
	`signature`	The signature relating to the exception.
	`tciPars`	The signature parameters relating to the exception.
	`excValue`	The exception to be sent.
	`transmissionFailure`	The failure message which might occur at transmission.
Return Value	`void`	
Constraint	Shall be called by CH to log a multicast raise operation. This event occurs after reply execution. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.51 tliPrCatchDetected_m

Signature	`void tliPrCatchDetected_m(in TString am, in TInteger ts,`
	`                          in TString src, in TInteger line,`
	`                          in TriComponentIdType c,`
	`                          in TriPortIdType at,`
	`                          in TriPortIdType from,`
	`                          in TriSignatureIdType signature,`
	`                          in TriExceptionType exc,`
	`                          in TriAddressType address)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is received.
	`from`	The port from which the exception has been sent.
	`signature`	The signature relating to the exception.
	`exc`	The catched exception.
	`address`	The address of the source within the SUT.

Return Value	`void`
Constraint	Shall be called by SA to log the catch enqueue operation. This event occurs after catch is enqueued. This event is used for logging the communication with the SUT.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.52 tliPrCatchDetected_c

Signature	`void tliPrCatchDetected_c(in TString am, in TInteger ts,`
	`                           in TString src, in TInteger line,`
	`                           in TriComponentIdType c,`
	`                           in TriPortIdType at,`
	`                           in TriPortIdType from,`
	`                           in TriSignatureIdType signature,`
	`                           in Value excValue)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is received.
	`from`	The port from which the exception has been sent.
	`signature`	The signature relating to the exception.
	`excValue`	The catched exception.
Return Value	`void`	
Constraint	Shall be called by CH to log the catch enqueue operation. This event occurs after catch is enqueued. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.53 tliPrCatchMismatch_m

Signature	`void tliPrCatchMismatch_m(in TString am, in TInteger ts,`
	`                           in TString src, in TInteger line,`
	`                           in TriComponentIdType c,`
	`                           in TriPortIdType at,`
	`                           in TriSignatureIdType signature,`
	`                           in Value excValue,`
	`                           in TciValueTemplate excTmpl,`
	`                           in TciValueDifferenceList diffs,`
	`                           in TriAddressType address,`
	`                           in TciValueTemplate addressTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.

In Parameters	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
	excValue	The received exception.
	excTmpl	The template used to check the exception match.
	diffs	The difference/the mismatch between exception and template
	address	The address of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a catch operation. This event occurs after catch is checked against a template. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.54 tliPrCatchMismatch_c

Signature	void tliPrCatchMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
	excValue	The received exception.
	excTmpl	The template used to check the exception match.
	diffs	The difference/the mismatch between exception and template
	from	The component which sent the reply.
	fromTmpl	The expected replying component.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a catch operation. This event occurs after catch is checked against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.55 tliPrCatch_m

Signature	`void tliPrCatch_m(in TString am, in TInteger ts,`
	`in TString src, in TInteger line,`
	`in TriComponentIdType c,`
	`in TriPortIdType at,`
	`in TriSignatureIdType signature,`
	`in Value excValue,`
	`in TciValueTemplate excTmpl,`
	`in TriAddressType address,`
	`in TciValueTemplate addressTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is received.
	`signature`	The signature relating to the exception.
	`excValue`	The received exception.
	`excTmpl`	The template used to check the exception match.
	`address`	The address of the source within the SUT.
	`addressTmpl`	The expected address of the source within the SUT.
Return Value	`void`	
Constraint	Shall be called by SA to log catching an exception. This event occurs after catch is checked against a template. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.56 tliPrCatch_c

Signature	`void tliPrCatch_c(in TString am, in TInteger ts,`
	`in TString src, in TInteger line,`
	`in TriComponentIdType c,`
	`in TriPortIdType at,`
	`in TriSignatureIdType signature,`
	`in Value excValue,`
	`in TciValueTemplate excTmpl,`
	`in TriComponentIdType from,`
	`in TciNonValueTemplate fromTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`at`	The port via which the exception is received.
	`signature`	The signature relating to the exception.
	`excValue`	The received exception.

In Parameters	excTmpl	The template used to check the exception match.
	from	The component which sent the reply.
	fromTmpl	The expected replying component.
Return Value	void	
Constraint	Shall be called by CH to log catching an exception. This event occurs after catch is checked against a template. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.57 tliPrCatchTimeoutDetected

Signature	void tliPrCatchTimeoutDetected(in TString am, in TInteger ts,
	                              in TString src,
	                              in TInteger line,
	                              in TriComponentIdType c,
	                              in TriPortIdType at,
	                              in TriSignatureIdType
	                              signature)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
Return Value	void	
Constraint	Shall be called by PA to log the detection of a catch timeout. This event occurs after the timeout is enqueued.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.58 tliPrCatchTimeout

Signature	void tliPrCatchTimeout (in TString am, in TInteger ts,
	                            in TString src,
	                            in TInteger line,
	                            in TriComponentIdType c,
	                            in TriPortIdType at,
	                            in TriSignatureIdType signature)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
Return Value	void	

Constraint	Shall be called by TE to log catching a timeout. This event occurs after the catch timeout has been performed.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.59 tliCCreate

Signature	`void tliCCreate(in TString am, in TInteger ts,`
	 `in TString src,`
	 `in TInteger line, in TriComponentIdType c,`
	 `in TriComponentIdType comp, in TString name,`
	 `in TBoolean alive)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`comp`	The component which is created.
	`name`	The name of the component which is created.
	`alive`	If the component is an alive component.
Return Value	`void`	
Constraint	Shall be called by TE to log the create component operation. This event occurs after component creation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.60 tliCStart

Signature	`void tliCStart(in TString am, in TInteger ts, in TString src,`
	 `in TInteger line, in TriComponentIdType c,`
	 `in TriComponentIdType comp,`
	 `in TciBehaviourIdType beh,`
	 `in TciParameterListType tciPars)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`comp`	The component which is started.
	`beh`	The behaviour being started on the component.
	`tciPars`	The parameters of the started behaviour.
Return Value	`void`	
Constraint	Shall be called by TE to log the start component operation. This event occurs after component start.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.61 tliCRunning

Signature	void tliCRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in ComponentStatusType status)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The component which is checked to be running.
	status	The status of this component.
Return Value	void	
Constraint	Shall be called by TE to log the running component operation. This event occurs after component running.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.62 tliCAlive

Signature	void tliCAlive(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in ComponentStatusType status)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The component which is checked to be running.
	status	The status of this component.
Return Value	void	
Constraint	Shall be called by TE to log the alive component operation. This event occurs after component alive.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.63 tliCStop

Signature	void tliCStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The component which is stopped.
Return Value	void	
Constraint	Shall be called by TE to log the stop component operation. This event occurs after component stop.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.64 tliCKill

Signature	void tliCKill(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The component which is killed.
Return Value	void	
Constraint	Shall be called by TE to log the kill component operation. This event occurs after component kill.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.65 tliCDoneMismatch

Signature	void tliCDoneMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The first component that did not match.
	compTmpl	The template used to check the done match.
Return Value	void	
Constraint	Shall be called by TE to log the mismatch of a done component operation. This event occurs after done is checked against a template.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.66 tliCDone

Signature	`void tliCDone (in TString am, in TInteger ts, in TString src,` `in TInteger line, in TriComponentIdType c,` `in TciNonValueTemplate compTmpl)`
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	compTmpl	The template used to check the done match.
Return Value	void	
Constraint	Shall be called by TE to log the done component operation. This event occurs after the done operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.67 tliCKilledMismatch

Signature	`void tliCKilledMismatch(in TString am, in TInteger ts,` `in TString src, in TInteger line,` `in TriComponentIdType c,` `in TriComponentIdType comp,` `in TciNonValueTemplate compTmpl)`
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	comp	The first component that did not match.
	compTmpl	The template used to check the killed match.
Return Value	void	

Constraint	Shall be called by TE to log the mismatch of a killed component operation. This event occurs after killed is checked against a template.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.68 tliCKilled

Signature	void tliCKilled (in TString am, in TInteger ts,
	                 in TString src, in TInteger line,
	                 in TriComponentIdType c,
	                 in TciNonValueTemplate compTmpl)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	compTmpl	The template used to check the killed match.
Return Value	void	
Constraint	Shall be called by TE to log the killed component operation. This event occurs after the killed operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.69 tliCTerminated

Signature	void tliCTerminated(in TString am, in TInteger ts,
	                    in TString src, in TInteger line,
	                    in TriComponentIdType c,
	                    in VerdictValue verdict)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	verdict	The verdict of the component.
Return Value	void	
Constraint	Shall be called by TE to log the termination of a component. This event occurs after the termination of the component.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.70 tliPConnect

Signature	void tliPConnect(in TString am, in TInteger ts, in TString src,
	                 in TInteger line, in TriComponentIdType c,
	                 in TriPortIdType port1, in TriPortIdType port2)

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port1	The first port to be connected.
	port2	The second port to be connected.
Return Value	void	
Constraint	Shall be called by CH to log the connect operation. This event occurs after the connect operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.71 tliPDisconnect

Signature	``void tliPDisconnect(in TString am, in TInteger ts,``
	``                     in TString src, in TInteger line,``
	``                     in TriComponentIdType c,``
	``                     in TriPortIdType port1,``
	``                     in TriPortIdType port2)``	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port1	The first port to be disconnected.
	port2	The second port to be disconnected.
Return Value	void	
Constraint	Shall be called by CH to log the disconnect operation. This event occurs after the disconnect operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.72 tliPMap

Signature	``void tliPMap(in TString am, in TInteger ts, in TString src,``
	``              in TInteger line, in TriComponentIdType c,``
	``              in TriPortIdType port1, in TriPortIdType port2)``	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port1	The first port to be mapped.
	port2	The second port to be mapped.
Return Value	void	

Constraint	Shall be called by SA to log the map operation. This event occurs after the map operation.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.73   tliPUnmap

Signature	`void tliPUnmap(in TString am, in TInteger ts, in TString src,`
	`                in TInteger line, in TriComponentIdType c,`
	`                in TriPortIdType port1, in TriPortIdType port2)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port1	The first port to be unmapped.
	port2	The second port to be unmapped.
Return Value	`void`	
Constraint	Shall be called by SA to log the unmap operation. This event occurs after the unmap operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.74   tliPClear

Signature	`void tliPClear(in TString am, in TInteger ts, in TString src,`
	`                in TInteger line, in TriComponentIdType c,`
	`                in TriPortIdType port)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port	The port to be cleared.
Return Value	`void`	
Constraint	Shall be called by TE to log the port clear operation. This event occurs after the port clear operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.75 tliPStart

Signature	void tliPStart(in TString am, in TInteger ts, in TString src,
	            in TInteger line, in TriComponentIdType c,
	            in TriPortIdType port)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port	The port to be started.
Return Value	void	
Constraint	Shall be called by TE to log the port start operation. This event occurs after the port start operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.76 tliPStop

Signature	void tliPStop(in TString am, in TInteger ts, in TString src,
	            in TInteger line, in TriComponentIdType c,
	            in TriPortIdType port)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port	The port to be stopped.
Return Value	void	
Constraint	Shall be called by TE to log the port stop operation. This event occurs after the port stop operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.77 tliPHalt

Signature	void tliPHalt(in TString am, in TInteger ts, in TString src,
	            in TInteger line, in TriComponentIdType c,
	            in TriPortIdType port)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port	The port to be stopped.
Return Value	void	

Constraint	Shall be called by TE to log the port halt operation. This event occurs after the port halt operation.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.78   tliEncode

Signature	`void tliEncode(in TString am, in TInteger ts, in TString src,`
	`                in TInteger line, in TriComponentIdType c,`
	`                in Value val, in TciStatusType encoderFailure,`
	`                in TriMessageType msg, in TString codec)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	value	The value to be encoded.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded value.
	codec	The used encoder.
Return Value	`void`	
Constraint	Shall be called by CD to log the encode operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.79   tliDecode

Signature	`void tliDecode(in TString am, in TInteger ts, in TString src,`
	`                in TInteger line, in TriComponentIdType c,`
	`                in Value val, in TciStatusType decoderFailure,`
	`                in TriMessageType msg, in TString codec)`	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	msg	The value to be decoded.
	decoderFailure	The failure message which might occur at decoding.
	value	The decoded value.
	codec	The used decoder.
Return Value	`void`	
Constraint	Shall be called by CD to log the decode operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.80 tliTTimeoutDetected

Signature	`void tliTTimeoutDetected(in TString am, in TInteger ts,`
	`                           in TString src, in TInteger line,`
	`                           in TriComponentIdType c,`
	`                           in TriTimerIdType timer)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`timer`	The timer that timed out.
Return Value	`void`	
Constraint	Shall be called by PA to log the detection of a timeout. This event occurs after timeout is enqueued.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.81 tliTTimeoutMismatch

Signature	`void tliTTimeoutMismatch(in TString am, in TInteger ts,`
	`                           in TString src, in TInteger line,`
	`                           in TriComponentIdType c,`
	`                           in TriTimerIdType timer,`
	`                           in TciNonValueTemplate timerTmpl)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`timer`	The first timer that did not match.
	`timerTmpl`	The timer template that did not match.
Return Value	`void`	
Constraint	Shall be called by TE to log a timeout mismatch. This event occurs after a timeout match failed.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.82 tliTTimeout

Signature	`void tliTTimeout(in TString am, in TInteger ts,`
	`                   in TString src, in TInteger line,`
	`                   in TriComponentIdType c,`
	`                   in TriTimerIdType timer,`
	`                   in TciNonValueTemplate timerTmpl)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	timer	The timer that matched.
	timerTmpl	The timer template that matched.
Return Value	void	
Constraint	Shall be called by TE to log a timeout match. This event occurs after a timeout matched.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.83 tliTStart

Signature	void tliTStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)	
Return Value		
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	timer	The timer that is started.
	dur	The timer duration.
Return Value	void	
Constraint	Shall be called by PA to log the start of a timer. This event occurs after the start timer operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.84 tliTStop

Signature	void tliTStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	timer	The timer that is stopped.
	dur	The duration of the timer when it was stopped.
Return Value	void	

Constraint	Shall be called by PA to log the stop of a timer. This event occurs after the stop timer operation.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.85 tliTRead

Signature	`void tliTRead(in TString am, in TInteger ts, in TString src,` `in TInteger line, in TriComponentIdType c,` `in TriTimerIdType timer,` `in TriTimerDurationType elapsed)`	
In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`timer`	The timer that is started.
	`elapsed`	The elapsed time of the timer.
Return Value	`void`	
Constraint	Shall be called by PA to log the reading of a timer. This event occurs after the read timer operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.86 tliTRunning

Signature	`void tliTRunning(in TString am, in TInteger ts,` `in TString src, in TInteger line,` `in TriComponentIdType c,` `in TriTimerIdType timer,` `in TimerStatusType status)`	
In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`timer`	The timer which is checked to be running.
	`status`	The status of this component.
Return Value	`void`	
Constraint	Shall be called by PA to log the running timer operation. This event occurs after the running timer operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.87   tliSEnter

Signature	void tliSEnter(in TString am, in TInteger ts, in TString src,
	                in TInteger line, in TriComponentIdType c,
	                in QualifiedName name,
	                in TciParameterListType tciPars,
	                in TString kind)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the scope.
	tciPars	The parameters of the scope.
	kind	The kind of the scope.
Return Value	void	
Constraint	Shall be called by TE to log the entering of a scope. This event occurs after the scope has been entered.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.88   tliSLeave

Signature	void tliSLeave(in TString am, in TInteger ts, in TString src,
	                in TInteger line, in TriComponentIdType c,
	                in QualifiedName name,
	                in TciParameterListType tciPars,
	                in Value returnValue, in TString kind)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the scope.
	tciPars	The parameters of the scope.
	returnValue	The return value of the scope.
	kind	The kind of the scope.
Return Value	void	
Constraint	Shall be called by TE to log the leaving of a scope. This event occurs after the scope has been left.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.89 tliVar

Signature	void tliVar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value varValue)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the variable.
	varValue	The new value of the variable.
Return Value	void	
Constraint	Shall be called by TE to log the modification of the value of a variable. This event occurs after the values have been changed.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.90 tliModulePar

Signature	void tliModulePar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value parValue)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the module parameter.
	parValue	The value of the module parameter.
Return Value	void	
Constraint	Shall be called by TE to log the value of a module parameter. This event occurs after the access to the value of a module parameter.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.91 tliGetVerdict

Signature	void tliGetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	verdict	The current value of the local verdict.
Return Value	void	
Constraint	Shall be called by TE to log the getverdict operation. This event occurs after the getverdict operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.92 tliSetVerdict

Signature	void tliSetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	verdict	The value to be set to the local verdict.
Return Value	void	
Constraint	Shall be called by TE to log the setverdict operation. This event occurs after the setverdict operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.93 tliLog

Signature	void tliLog (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString log)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	log	The string to be logged.
Return Value	void	
Constraint	Shall be called by TM to log the TTCN-3 statement log. This event occurs after the TTCN-3 log operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.94 tliAEnter

Signature	void tliAEnter(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log entering an alt. This event occurs after an alt has been entered.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.95 tliALeave

Signature	void tliALeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log leaving an alt. This event occurs after the alt has been left.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.96 tliANomatch

Signature	void tliANomatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log the nomatch of an alt. This event occurs after the alt has not matched.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.97 tliARepeat

Signature	void tliARepeat(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log repeating an alt. This event occurs when the alt is being repeated.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.98 tliADefaults

Signature	void tliADefaults(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log entering the default section. This event occurs after the default section has been entered.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.99 tliAActivate

Signature	void tliAActivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars, in Value ref)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the default.
	tciPars	The parameter of the default.
	ref	The resulting default reference.
Return Value	void	

Constraint	Shall be called by TE to log the activation of a default. This event occurs after the default activation.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.100 tliADeactivate

Signature	`void tliADeactivate(in TString am, in TInteger ts,` `                    in TString src, in TInteger line,` `                    in TriComponentIdType c,` `                    in Value ref)`	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	ref	The resulting default reference.
Return Value	void	
Constraint	Shall be called by TE to log the deactivation of a default. This event occurs after the default deactivation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.101 tliAWait

Signature	`void tliAWait(in TString am, in TInteger ts, in TString src,` `              in TInteger line, in TriComponentId c)`	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
Return Value	void	
Constraint	Shall be called by TE to log that the component awaits events for a new snapshot.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.102 tliAction

Signature	`void tliAction(in TString am, in TInteger ts, in TString src,` `               in TInteger line, in TriComponentIdType c,` `               in TString action)`

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	action	The action to be performed.
Return Value	void	
Constraint	Shall be called by TE to log that the component executed an SUT action.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.103 tliMatch

Signature	void tliMatch(in TString am, in TInteger ts, in TString src,
	                in TInteger line, in TriComponentIdType c,
	                in Value expr, in TciValueTemplate tmpl)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	expr	The expression to be matched with tmpl.
	tmpl	The template to be matched with expr.
Return Value	void	
Constraint	Shall be called by TE to log that the component successfully executed a match operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

7.3.4.1.104 tliMatchMismatch

Signature	void tliMatchMismatch(in TString am, in TInteger ts,
	                        in TString src, in TInteger line,
	                        in TriComponentIdType c, in Value expr,
	                        in TciValueTemplate tmpl,
	                        in TciValueDifferenceList diffs)	

In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	expr	The expression to be matched with tmpl
	tmpl	The template to be matched with expr
	diffs	The difference/the mismatch between expr and tmpl
Return Value	void	

Constraint	Shall be called by TE to log that the component unsuccessfully executed a match operation – a mismatch occurred.
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.

7.3.4.1.105 tliInfo

Signature	`void tliInfo (in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TInteger level, in TString info)`	

In Parameters	`am`	An additional message.
	`ts`	The time when the event is produced.
	`src`	The source file of the test specification.
	`line`	The line number where the request is performed.
	`c`	The component which produces this event.
	`level`	The information level.
	`info`	The information.
Return Value	`void`	
Constraint	Can be called by TE to log additional information during test execution. This generation of this event is tool dependent as well as the usage of the parameters level and info.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of this Recommendation.	

8   Java language mapping

8.1   Introduction

This clause introduces the TCI Java language mapping. For efficiency reasons a dedicated language mapping is introduced instead of using the OMG IDL to Java language.

The Java language mapping for the TTCN-3 Control Interface defines how the IDL definitions described in clause 7 are mapped to the Java language. The language mapping is independent of the used Java version as only basic Java language constructs are used.

8.2   Names and scopes

8.2.1   Names

Although there are no conflicts between identifiers used in the IDL definition and the Java language, some naming translation rules are applied to the IDL identifiers.

Java interfaces or class identifiers are omitting the trailing `Type` used in the IDL definition.

Example: The IDL type **`TciTestCaseIdType`** maps to `TciTestCaseId` in Java.

The resulting mapping conforms to the standard Java coding conventions.

8.2.2   Scopes

The IDL module **`tciInterface`** is mapped to the Java package `org.etsi.ttcn3.tci`. All IDL type declarations within this module are mapped to Java classes or interface declarations within this package.

Type mapping

Basic type mapping

Table 2 gives an overview on how the native types TBoolean, TFloat, TInteger, TString, and TStringSeq are mapped to the Java types.

Table 2 – Basic type mappings

IDL Type	Java Type
TBoolean	boolean
TFloat	float
TInteger	int
TString	java.lang.String
TStringSeq	java.lang.String[]

The native type TObjId is defined in the respective section of the ObjidValue interface.

boolean

The IDL `TBoolean` type is mapped to the java basic type `boolean`.

float

The IDL `TFloat` type is mapped to the java basic type `float`.

char

The IDL `TChar` type is mapped to the java basic type `char`.

int

The IDL `TInteger` type is mapped to the java basic type `int`.

String

The IDL `TString` type is mapped to the `java.lang.String` class without range checking or bounds for characters in the string. All possible strings defined in TTCN-3 can be converted to `java.lang.String`.

String[]

The IDL `TStringSeq` type is mapped to an array of the `java.lang.String` class.

Universal Char

The IDL `TUniversalChar` type is mapped to a java basic type `int`. The integer uses the canonical form as defined in ISO/IEC 10646, clause 6.2.

Structured type mapping

The TCI IDL description defines user-defined types as native types. In the Java language mapping these types are mapped to Java interfaces. The interfaces define methods and attributes being available for objects implementing this interface.

8.2.2.1   TciParameterType

`TciParameterType` is mapped to the following interface:

```
// TCI IDL TciParameterType
package org.etsi.ttcn.tci;
public interface TciParameter {
   public String   getParameterName();
```

```
   public void       setParameterName(String name);
   public int        getParameterPassingMode();
   public void       setParameterPassingMode(TciParameterPassingMode mode);
   public Value      getParameter();
   public void       setParameter(Value parameter);
}
```

Methods:

- `getParameterName ()`   Returns the parameter name as defined in the TTCN-3 specification.

- `setParameterName (String name)` Sets the name of this `TciParameter` parameter to `name`.

- `getParameterPassingMode()`   Returns the parameter passing mode of this parameter.

- `setParameterPassingMode(TciParameterPassingMode mode)`
                                     Sets the parameter mode of this `TriParameter` parameter to `mode`.

- `getParameter()`   Returns the Value parameter of this `TciParameter`, or the `null` object if the parameter contains the distinct value `null`.

- `setParameter(Value parameter)` Sets the Value parameter of this `TciParameter` to parameter. If the distinct value null shall be set to indicate that this parameter holds no value, the Java `null` shall be passed as parameter.

8.2.2.2   TciParameterPassingModeType

TciParameterPassingModeType is mapped to the following interface:

```
// TCI IDL TciParameterPassingModeType
package org.etsi.ttcn.tci;
public interface TciParameterPassingMode {
   public final static int TCI_IN     = 0;
   public final static int TCI_INOUT  = 1;
   public final static int TCI_OUT    = 2;
}
```

Constants:

- `TCI_IN`   Will be used to indicate that a `TciParameter` is an `in` parameter.

- `TCI_INOUT`   Will be used to indicate that a `TciParameter` is an `inout` parameter.

- `TCI_OUT`   Will be used to indicate that a `TciParameter` is an `out` parameter.

8.2.2.3   TciParameterListType

TciParameterListType is mapped to the following interface:

```
// TCI IDL TciParameterListType
package org.etsi.ttcn.tci;
public interface TciParameteList {
   public int                   size() ;
   public boolean               isEmpty() ;
   public java.util.Enumeration getParameters() ;
   public TciParameter          get(int index) ;
   public void                  clear() ;
   public void                  add(TciParameter parameter) ;
   public void                  setParameters(TciParameter[] parameters) ;
}
```

Methods:

- size()   Returns the number of parameters in this list.

- isEmpty()   Returns true if this list contains no parameters.

- getParameters()   Returns an Enumeration over the parameters in the list. The enumeration provides the parameters in the same order as they appear in the list.

- get(int index)   Returns the TciParameter at the specified position.

- clear()   Removes all parameters from this TciParameterList.

- add(TciParameter parameter)

                                      Adds parameter to the end of this TciParameterList.

- setParameter(TciParameter[] parameters)

                                      Fills this TciParameterList with parameters.

8.2.2.4   TciTypeClassType

TciTypeClassType is mapped to the following interface:

```
// TCI IDL TciTypeClassType
package org.etsi.ttcn.tci;
public interface TciTypeClass {
    public final static int ADDRESS             = 0 ;
    public final static int ANYTYPE             = 1 ;
    public final static int BITSTRING           = 2 ;
    public final static int BOOLEAN             = 3 ;
    public final static int CHARSTRING          = 5 ;
    public final static int COMPONENT           = 6 ;
    public final static int ENUMERATED          = 7 ;
    public final static int FLOAT               = 8 ;
    public final static int HEXSTRING           = 9 ;
    public final static int INTEGER             = 10 ;
    public final static int OBJID               = 11 ;
    public final static int OCTETSTRING         = 12 ;
    public final static int RECORD              = 13 ;
    public final static int RECORD_OF           = 14 ;
    public final static int SET                 = 15 ;
    public final static int SET_OF              = 16 ;
    public final static int UNION               = 17 ;
    public final static int UNIVERSAL_CHARSTRING = 19 ;
    public final static int VERDICT             = 20 ;
}
```

8.2.2.5   TciTestComponentKindType

TciTestComponentKindType is mapped to the following interface:

```
// TCI IDL TciTestComponentKindType
public interface TciTestComponentKind {
    public final static int TCI_CTRL_COMP    = 0;
    public final static int TCI_MTC_COMP     = 1;
    public final static int TCI_PTC_COMP     = 2;
    public final static int TCI_SYSTEM_COMP  = 3;
    public final static int TCI_ALIVE_COMP   = 4;
}
```

8.2.2.6   TciBehaviourIdType

TciBehaviourIdType is mapped to the following interface:

```
// TCI IDL TciBehaviourIdType
package org.etsi.ttcn.tci;
```

```
public interface TciBehaviourId extends QualifiedName {
}
```

8.2.2.7   TciTestCaseIdType

`TciTestCaseIdType` is mapped to the following interface:

```
// TCI IDL TciTestCaseIdType
package org.etsi.ttcn.tci;
public interface TciTestCaseId  extends QualifiedName {
}
```

8.2.2.8   TciModuleIdType

`TciModuleIdType` is mapped to the following interface:

```
// TCI IDL TciModuleIdType
package org.etsi.ttcn.tci;
public interface TciModuleId  extends QualifiedName {
}
```

8.2.2.9   TciModuleParameterIdType

`TciModuleParameterIdType` is mapped to the following interface:

```
// TCI IDL TciModuleParameterIdType
package org.etsi.ttcn.tci;
public interface TciModuleParameterId  extends QualifiedName {
}
```

8.2.2.10   TciModuleParameterListType

`TciModuleParameterListType` is mapped to the following interface:

```
// TCI IDL TciModuleParameterListType
package org.etsi.ttcn.tci;
public interface TciModuleParameterList {
   public int                    size() ;
   public boolean                isEmpty() ;
   public java.util.Enumeration getModuleParameters() ;
   public TciModuleParameter    get(int index) ;
}
```

Methods:

- size()   Returns the number of module parameters in this list.

- isEmpty()   Returns `true` if this list `contains` no module parameters.

- getModuleParameters()   Returns an `Enumeration` over the module parameters in the list. The enumeration provides the module parameters in the same order as they appear in the list.

- get(int index)   Returns the `TciModuleParameter` at the specified position.

8.2.2.11   TciModuleParameterType

`TciModuleParameterType` is mapped to the following interface:

```
// TCI IDL TciModuleParameterType
package org.etsi.ttcn.tci;
public interface TciModuleParameter {
```

```
    public String       getModuleParameterName();
    public Value     getDefaultValue();
}
```

Methods:

- `getModuleParameterName ()`

  Returns the module parameter name as defined in the TTCN-3 specification.

- `getDefaultValue()` Returns the default Value module parameter of this `TciModuleParameter`, or the `null` object if the module parameter contains the distinct value `null`.

8.2.2.12   TciParameterTypeListType

`TciParameterTypeListType` is mapped to the following interface:

```
// TCI IDL TciParameterTypeListType
package org.etsi.ttcn.tci;
public interface TciParameterTypeList {
    public int                   size() ;
    public boolean               isEmpty() ;
    public java.util.Enumeration getParameterTypes() ;
    public TciParameterType      get(int index) ;
}
```

Methods:

- `size()`   Returns the number of parameter types in this list.

- `isEmpty()`   Returns `true` if this list contains no parameter types.

- `getParameterTypes()`   Returns an `Enumeration` over the parameter types in the list. The enumeration provides the parameter types in the same order as they appear in the list.

- `get(int index)`   Returns the `TciParameterType` at the specified position.

8.2.2.13   TciModuleIdListType

`TciModuleIdListType` is mapped to the following interface:

```
// TCI IDL TciModuleIdListType
package org.etsi.ttcn.tci;
public interface TciModuleIdList {
    public int                   size() ;
    public boolean               isEmpty() ;
    public java.util.Enumeration getImportedModules() ;
    public TciModuleId           get(int index) ;
}
```

Methods:

- `size()`   Returns the number of modules in this list.

- `isEmpty()`   Returns `true` if this list contains no modules.

- `getImportedModules()`   Returns an `Enumeration` over the modules in the list. The enumeration provides the modules in the same order as they appear in the list.

- `get(int index)`   Returns the `TciModuleId` at the `specified` position.

8.2.3   Abstract type mapping

The TTCN-3 data types are modelled in Java using the abstract type mapping as defined in this clause. The `Type` interface defines only operations used to retrieve in TTCN-3 defined types. No TTCN-3 types can be constructed using the `Type` interface. Types are modelled using the single interface `Type`, that provides methods to identify types and to retrieve values of a given type.

8.2.3.1   Type

`Type` is mapped to the following interface:

```
// TCI IDL Type
package org.etsi.ttcn.tci;
public interface Type {
    public TciModuleId getDefiningModule ();
    public String      getName ();
    public int         getTypeClass ();
    public Value       newInstance ();
    public String      getTypeEncoding ();
    public String      getTypeEncodingVariant();
    public String[]    getTypeExtension();
}
```

Methods:

- `getDefiningModule()`   Returns the module identifier of the module the type has been defined in. If the type represents a TTCN-3 base type the distinct value `null` will be returned.

- `getName()`   Returns name of the type as defined in the TTCN-3 module.

- `getTypeClass()`   Returns the type class of the respective type. A value of `TciTypeClassType` can have one of the following constants: ADDRESS, ANYTYPE, BITSTRING, BOOLEAN, CHARSTRING, COMPONENT, ENUMERATED, FLOAT, HEXSTRING, INTEGER, OBJID, OCTETSTRING, RECORD, RECORD_OF, SET, SET_OF, UNION, UNIVERSAL_CHARSTRING, VERDICT.

- `newInstance()`   Returns a freshly created value of the given type. This initial value of the created value is undefined.

- `getTypeEncoding()`   Returns the type encoding attribute as defined in the TTCN-3 module.

- `getTypeEncodingVariant()` This operation returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute has been defined the distinct value `null` will be returned.

- `getTypeExtension()`   Returns the type extension attribute as defined in the TTCN-3 module.

8.2.4   Abstract value mapping

TTCN-3 values can be retrieved from the TE and constructed using the Value interface. The value mapping interface is constructed hierarchically with Value as the basic interface. Specialized interfaces for different types of values have been defined.

8.2.4.1 Value

Value is mapped to the following interface:

```
// TCI IDL Value
package org.etsi.ttcn.tci;
public interface Value {
    public Type      getType() ;
    public boolean   notPresent() ;
    public String    getValueEncoding() ;
    public String    getValueEncodingVariant();
}
```

Methods:

- getType()   Returns the type of the specified value.

- notPresent()   Returns `true` if the specified value is `omit`, `false` otherwise.

- getValueEncoding()   This operation returns the value encoding attribute as defined in TTCN-3, if any. If no encoding attribute has been defined the distinct value `null` will be returned.

- getValueEncodingVariant()

                               This operation returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute has been defined the distinct value `null` will be returned.

8.2.4.2 IntegerValue

IntegerValue type is mapped to the following interface:

```
// IntegerValue
package org.etsi.ttcn.tci;
public interface IntegerValue {
    public void      setInteger(int value);
    public int       getInteger();
}
```

Methods:

- setInteger(int value)   Sets this IntegerValue to the int value `value`.

- getInteger()   Returns the int value represented by this IntegerValue.

8.2.4.3 FloatValue

FloatValue type is mapped to the following interface:

```
// FloatValue
package org.etsi.ttcn.tci;
public interface FloatValue {
    public void      setFloat(float value);
    public float getFloat();
}
```

Methods:

- setFloat(float value)   Sets this FloatValue to the float value `value`.

- getFloat()   Returns the float value represented by this FloatValue.

8.2.4.4 BooleanValue

`BooleanValue` type is mapped to the following interface:

```
// BooleanValue
package org.etsi.ttcn.tci;
public interface BooleanValue {
    public void     setBoolean(boolean value);
    public boolean  getBoolean();
}
```

Methods:

- `setBoolean(boolean value)`   Sets this BooleanValue to the boolean value `value`.

- `getBoolean()`   Returns the boolean value represented by this BooleanValue.

8.2.4.5 ObjidValue

`ObjidValue` is mapped to the following interface:

```
// TCI IDL ObjidValue
package org.etsi.ttcn.tci;
public interface ObjidValue {
    TciObjId  getObjid ();
    void      setObjid (TciObjId value);
}
```

Methods:

- `getObjid()`   Returns the object id value of the TTCN-3 `objid`.

- `setObjid(TciObjId value)`   Sets this `ObjidValue` to `value`.

8.2.4.6 TciObjId

`TciObjId` is mapped to the following interface. The native java representation of a TTCN-3 ObjectId consists of an ordered sequence of `TciObjIdElements`.

```
package org.etsi.ttcn.tci;
public interface TciObjId {
    public int              size() ;
    public void             setObjElement(TciObjIdElement[] objElements) ;
    public TciObjIdElement getObjElement(int index) ;
}
```

Methods:

- `size()`   Returns the size of this Object Id in `TciObjIdElements`.

- `setObjElement(TciObjIdElement[] objElements)`
  Sets this ObjId to the list of objElements.

- `getObjElement(int index)`   Returns the `TciObjIdElement` at position `index`.

8.2.4.7 TciObjIdElement

A TciObjIdElement represents a single object element within a TTCN-3 ObjId value. It can be set using different representations like the ASCII representation or as integer.

`TciObjIdElement` is mapped to the following interface:

```
package org.etsi.ttcn.tci;
public interface TciObjIdElement {
```

```
    public void       setElementAsAscii(String element) ;
    public void       setElementAsNumber(int element) ;
    public String     getElementAsAscii() ;
    public int        getElementAsNumber() ;
}
```

Methods:

•   `setElementAsAscii(String element)`

> Sets the internal representation of this ObjIdElement to string value `element`.

•   `setElementAsNumber(int element)`

> Sets the internal representation of this ObjIdElement to the integer value `element`.

•   `getElementAsAscii()`   Returns the internal representation of this ObjIdElement as string.

•   `getElementAsNumber()`   Returns the internal representation of this ObjIdElement as integer.

8.2.4.8 CharstringValue

CharstringValue is mapped to the following interface:

```
// TCI IDL CharstringValue
package org.etsi.ttcn.tci;
public interface CharstringValue {
    String getString ();
    void   setString (String value);
    char   getChar (int position);
    void   setChar (int  position, char value);
    int    getLength ();
    void   setLength (int len);
}
```

Methods:

•   `getString()`   Returns the string of the TTCN-3 `charstring`. The textual representation of the empty TTCN-3 `charstring` is `''`, while its length is zero.

•   `setString(String value)` Sets this `CharstringValue` to `value`.

•   `getChar(int position)`   Returns the char value of the TTCN-3 charstring at `position`. Position 0 denotes the first char of the TTCN-3 `charstring`. Valid values for position are `0` to `length - 1`.

•   `setChar(int position, char value)`

> Sets the char at `position` to `value`. Valid values for position are `0` to `length - 1`.

•   `getLength()`   Returns the length of this `CharstringValue` in chars, zero if the value of this `CharstringValue` is `omit`.

•   `setLength(int len)`   Sets the length of this `CharstringValue` in chars to `len`.

8.2.4.9 BitstringValue

BitstringValue is mapped to the following interface:

```
// TCI IDL BitstringValue
```

```
package org.etsi.ttcn.tci;
public interface BitstringValue {
    String getString ();
    void   setString (String value);
    int    getBit (int position);
    void   setBit (int  position, int value);
    int    getLength ();
    void   setLength (int len);
}
```

Methods:

•	getString()	Returns the textual representation of this `BitstringValue`, as defined in TTCN-3. E.g., the textual representation of `0101` is `"0101"B`. The textual representation of the empty TTCN-3 `bitstring` is `""B`, while its length is zero.

•   setString(String value) Sets the value of this `BitstringValue` according to the textual representation as defined by `value`. E.g., The value of this `BitstringValue` will be `0101` if the textual representation in value is `"0101"B`.

•   getBit(int position) Returns the value (`0 | 1`) at position of this TTCN-3 bitstring. Position 0 denotes the first bit of the TTCN-3 bitstring. Valid values for position are `0` to `length - 1`.

•   setBit(int position, int value)

     Sets the bit at position to value (`0 | 1`). Position 0 denotes the first bit in this `BitstringValue`. Valid values for position are `0` to `length - 1`.

•   getLength() Returns the length of this `BitstringValue` in bits, zero if the value of this `BitstringValue` is `omit`.

•   setLength(int len) Sets the length of this `BitstringValue` in bits to `len`.

8.2.4.10 OctetstringValue

OctetstringValue is mapped to the following interface:

```
// TCI IDL OctetstringValue
package org.etsi.ttcn.tci;
public interface OctetstringValue {
    String getString ();
    void   setString (String value);
    int    getOctet (int position);
    void   setOctet (int  position, int value);
    int    getLength ();
    void   setLength (int len);
}
```

Methods:

•   getString() Returns the textual representation of this `OctetstringValue`, as defined in TTCN-3. E.g., the textual representation of `0xCAFFEE` is `"CAFFEE"O`. The textual representation of the empty TTCN-3 `octetstring` is `""O`, while its length is zero.

•   setString(String value) Sets the value of this `OctetstringValue` according to the textual representation as defined by `value`. E.g., the value of

this `OctetstringValue` will be `0xCAFFEE` if the textual representation in value is `"CAFFEE"O`.

- `getOctet(int position)`   Returns the value (0..255) at `position` of this TTCN-3 octetstring. Position 0 denotes the first octet of the TTCN-3 octetstring. Valid values for position are `0` to `length - 1`.

- `setOctet(int position, int value)`

  Sets the octet at `position` to value (0..255). Position 0 denotes the first octet in the octetstring. Valid values for position are `0` to `length - 1`.

- `getLength()`   Returns the length of this `OctetstringValue` in octets, zero if the value of this `OctetstringValue` is `omit`.

- `setLength(int len)`   Sets the length of this `OctetstringValue` in octets to `len`.

8.2.4.11   UniversalCharstringValue

`UniversalCharstringValue` is mapped to the following interface:

```
// TCI IDL UniversalCharstringValue
package org.etsi.ttcn.tci;
public interface UniversalCharstringValue {
    String getString ();
    void   setString (String value);
    int    getChar (int position);
    void   setChar (int  position, char value);
    int    getLength ();
    void   setLength (int len);
}
```

Methods:

- `getString()`   Returns the textual representation of this `UniversalCharstringValue`, as defined in TTCN-3.

- `setString(String value)`   Sets the value of this `UniversalCharstringValue` according to the textual representation as defined by `value`.

- `getChar(int position)`   Returns the `UniversalChar` value of the TTCN-3 `universal charstring` at position. Position 0 denotes the first `UniversalChar` of the TTCN-3 `universal charstring`. Valid values for `position` are `0` to `length - 1`.

- `setChar(int position, char value)`

  Set the `UniversalChar` at `position` to `value`. Valid values for `position` are `0` to `length - 1`.

- `getLength()`   Returns the length of this `UniversalCharstringValue` in `UniversalChars`, zero if the value of this `UniversalCharstringValue` is `omit`.

- `setLength(int len)`   Sets the length of this `UniversalCharstringValue` in `UniversalChars` to `len`.

8.2.4.12   HexstringValue

`HexstringValue` is mapped to the following interface:

```
// TCI IDL HexstringValue
```

```
package org.etsi.ttcn.tci;
public interface HexstringValue {
    String getString ();
    void    setString (String value);
    int     getHex (int position);
    void    setHex (int  position, int value);
    int     getLength ();
    void    setLength (int len);
}
```

Methods:

- getString()   Returns the textual representation of this `HextstringValue`, as defined in TTCN-3. E.g., the textual representation of `0xAFFEE` is `"AFFEE"H`. The textual representation of the empty TTCN-3 `hexstring` is `""H`, while its length is zero.

- setString(String value)   Sets the value of this `HexstringValue` according to the textual representation as defined by `value`. E.g., the value of this `HexstringValue` will be `0xAFFEE` if the textual representation in `value` is `"AFFEE"H`.

- getHex(int position)   Returns the value `(0...15)` at `position` of this TTCN-3 hexstring. Position 0 denotes the first hex digits of the TTCN-3 hexstring. Valid values for position are `0` to `length – 1`.

- setHex(int position, int value)
                              Set the hex digit at position to value `(0...16)`. Position 0 denotes the first octet in the `hexstring`. Valid values for position are `0` to `length – 1`.

- getLength()   Returns the length of this `HexstringValue` in octets, zero if the value of this `HexstringValue` is `omit`.

- setLength(int len)   Sets the length of this `HexstringValue` in hex digits to len.

8.2.4.13   RecordValue

RecordValue is mapped to the following interface:

```
// TCI IDL RecordValue
package org.etsi.ttcn.tci;
public interface RecordValue {
    public Value getField(String fieldName) ;
    public void     setField(String fieldName, Value value) ;
    public String[] getFieldNames() ;
    public void  setFieldOmitted(String fieldName)
}
```

Methods:
- getField(String fieldName)

                              Returns the value of the field named `fieldName`. The return value is the common abstract base type `Value`, as a record field can have any type defined in TTCN-3. If the field cannot be obtained from the record the distinct value `null` will be returned.

- setField(String fieldName, Value value)

                              Sets the field named `fieldName` of the record to `value`. No assumption shall be made on how a field is stored in a record.

An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore it should be assumed that subsequent modifications of `value` will not be considered in the record.

- `getFieldNames()`   Returns an array of String of field names, the empty sequence, if the record has no fields.

- `setFieldOmitted(String fieldName)`

     Sets the field named `fieldName` of the record to `omit`.

8.2.4.14 RecordOfValue

`RecordOfValue` is mapped to the following interface:

```
// TCI IDL RecordOfValue
package org.etsi.ttcn.tci;
public interface RecordOfValue {
    public Value getField(String fieldName) ;
    public void      setField(int position, Value value) ;
    public void      appendField(Value value) ;
    public Type      getElementType() ;
    public int       getLength() ;
    public void      setLength(int len) ;
}
```

Methods:

- `getField(String fieldName)`

> Returns the value of the `record of` at `position` if `position` is between `zero` and `length -1`, the distinct value `null` otherwise. The return value is the common abstract base type `Value`, as a `record of` can have fields of any type defined in TTCN-3.

- `setField(int position, Value value)`

> Sets the field at `position` to `value`. If `position` is greater than `(length -1)` the `record of` will be extended to have the length `(position + 1)`. The `record of` elements between the original position at `length` and `position - 1` will be set to `OMIT`. No assumption shall be made on how a field is stored in a `record of`. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore, it should be assumed that subsequent modifications of `value` will not be considered in the `record of`.

- `appendField(Value value)`

> Appends the value at the end of the `record of`, i.e., at position `length`. No assumption shall be made on how a field is stored in a `record of`. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore, it should be assumed that subsequent modifications of `value` will not be considered in the `record of`.

- getElementType()   This operation will return the `Type` of the elements of this `record of`.

- getLength()   Returns the actual length of the `record of` value, zero if the `record of` value is OMIT.

- setLength(int len)   Sets the length of the `record of` to `len`. If `len` is greater than the original length, newly created elements have the value OMIT. If `len` is less than or equal to the original length this operation will be ignored.

8.2.4.15   UnionValue

UnionValue is mapped to the following interface:

```
// TCI IDL UnionValue
package org.etsi.ttcn.tci;
public interface UnionValue {
    Value    getVariant  (String variantName);
    void     setVariant  (String variantName, Value value);
    String   getPresentVariantName ();
    String[] getVariantNames ();
}
```

Methods:

- getVariant(String variantName)

    Returns the value of the TTCN-3 union `variantName`, if `variantName` equals the result of `getPresentVariantName`, the distinct value `null` otherwise. `variantName` denotes the name of the union variant as defined in TTCN-3.

- setVariant(String variantName, Value value)

    Sets `variantName` of the union to value. If `variantName` is not defined for this union, this operation will be ignored. If another variant was selected, the new variant will be selected instead.

- getPresentVariantName() Returns the variant name that has a value in this union set as a `String`. The distinct value `null` will be returned if no variant is selected.

- getVariantNames()   Returns an array of `String` of variant names, the empty sequence, if the union has no fields. If the `UnionValue` represents the TTCN-3 `anytype`, i.e., the type class of the type obtained by getType() is ANYTYPE, all predefined and user-defined TTCN-3 types will be returned.

8.2.4.16   EnumeratedValue

EnumeratedValue is mapped to the following interface:

```
// TCI IDL EnumeratedValue
package org.etsi.ttcn.tci;
public interface EnumeratedValue {
    String getEnum ();
    void   setEnum (String enumValue);
}
```

Methods:

- getEnum()   Returns the string identifier of this `EnumeratedValue`. This identifier equals the identifier in the TTCN-3 specification.

- setEnum(String enumValue)   Sets the enum to `enumValue`. If `enumValue` is not an allowed value for this enumeration the operation will be ignored.

8.2.4.17 VerdictValue

VerdictValue is mapped to the following interface:

```
// TCI IDL VerdictValue
package org.etsi.ttcn.tci;
public interface VerdictValue {
    public static final int NONE = 0;
    public static final int PASS = 1;
    public static final int INCONC  = 2;
    public static final int FAIL = 3;
    public static final int ERROR   = 4;

    public int      getVerdict() ;
    public void     setVerdict(int verdict) ;
}
```

Methods:

- getVerdict ()   Returns the integer value for this `VerdictValue`. The `integer` is one of the following constants: `ERROR`, `FAIL`, `INCONC`, `NONE`, `PASS`.

- setVerdict(int verdict)   Sets this `VerdictValue` to `verdict`. Note that a `VerdictValue` can be set to any of the above-mentioned verdicts at any time. The `VerdictValue` does not perform any verdict calculations as defined in TTCN-3. For example, it is legal to set the `VerdictValue` first to `ERROR` and then to `PASS`.

8.2.4.18 AddressValue

AddressValue is mapped to the following interface:

```
// TCI IDL AddressValue
package org.etsi.ttcn.tci;
public interface AddressValue {
    public int      getAddress() ;
    public void     setAddress(Value value) ;
}
```

Methods:

- getAddress()   Returns the value represented by this `AddressValue`.

- setAddress(Value value)   Sets this `AddressValue` to the value `value`.

8.2.5 Abstract logging types mapping

Additional types are defined to ease the logging of matches between values and templates.

8.2.5.1 TciValueTemplate

TciValueTemplate is mapped to the following interface:

```
// TCI IDL TciValueTemplate
package org.etsi.ttcn.tci;
public interface TciValueTemplate {
    public boolean isOmit();
    public boolean isAny();
    public boolean isAnyOrOmit();
    public String getTemplateDef();
}
```

Methods:

-      `isOmit()`   Returns `true` if the template is `omit`, `false` otherwise.

-      `isAny()`   Returns `true` if the template is `any`, `false` otherwise.

-      `isAnyOrOmit()`   Returns `true` if the template is `anyoromit`, `false` otherwise.

-      `getTemplateDef()` This operation returns the template definition.

8.2.5.2   TciNonValueTemplate

TciNonValueTemplate is mapped to the following interface:

```
// TCI IDL TciNonValueTemplate
package org.etsi.ttcn.tci;
public interface TciNonValueTemplate {
    public boolean isAny();
    public boolean isAll();
    public String getTemplateDef();
}
```

Methods:

-      `isAny()`   Returns `true` if the template is `any`, `false` otherwise.

-      `isAll()`   Returns `true` if the template is `all`, `false` otherwise.

-      `getTemplateDef()` This operation returns the template definition.

8.2.5.3   TciValueList

TciValueList is mapped to the following interface:

```
// TCI IDL TciValueList
package org.etsi.ttcn.tci;
public interface TciValueList{
    public int                  size() ;
    public boolean              isEmpty() ;
    public TciValue             get(int index) ;
}
```

Methods:

-      `size()`   Returns the number of values in this list.

-      `isEmpty()`   Returns `true` if this list contains no values.

-      `get(int index)`   Returns the `Value` at the specified position.

8.2.5.4   TciValueDifference

TciValueDifference is mapped to the following interface:

```
// TCI IDL TciValueDifference
package org.etsi.ttcn.tci;
public interface TciValueDifference {
    public Value getValue();
    public TciValueTemplate   getTciValueTemplate();
    public String   getDescription()
```

}

Methods:

- `getValue()`   Returns the value of this `TciValueDifference`.

- `getTciValueTemplate ()`   Returns the template of this `TciValueDifference`.

- `getDescription()`   Returns the description of the mismatch.

8.2.5.5   TciValueDifferenceList

`TciValueDifferenceList` is mapped to the following interface:

```
// TCI IDL TciValueDifferenceList
package org.etsi.ttcn.tci;
public interface TciValueDifferenceList{
    public int                  size() ;
    public boolean              isEmpty() ;
    public TciValueDifference   get(int index) ;
}
```

Methods:

- `size()`   Returns the number of differences in this list.

- `isEmpty()`   Returns `true` if this list contains no differences.

- `get(int index)`   Returns the `TciValueDifference` at the specified position.

8.3   Constants

Within this Java language mapping constants have been specified. All constants are defined `public final static` and are accessible from every object from every package. The constants defined within this clause are not defined with the IDL clause. Instead they result from the specification of the TCI IDL types marked as native.

The following constants can be used to determine the parameter passing mode of TTCN-3 parameters (see also clause 8.2.2.2).

- `org.etsi.ttcn.tci.TciParameterPassingMode.TCI_IN;`

- `org.etsi.ttcn.tci.TciParameterPassingMode.TCI_INOUT;`

- `org.etsi.ttcn.tri.TciParameterPassingMode.TCI_OUT;`

For the distinct parameter value `null`, the encoded parameter value shall be set to Java `null`.

The following constants shall be used for value handling (see also clause 8.2.2.4).

- `org.etsi.ttcn.tci.TciTypeClass.ADDRESS;`

- `org.etsi.ttcn.tci.TciTypeClass.ANYTYPE;`

- `org.etsi.ttcn.tci.TciTypeClass.BITSTRING;`

- `org.etsi.ttcn.tci.TciTypeClass.BOOLEAN;`

- `org.etsi.ttcn.tci.TciTypeClass.CHARSTRING;`

- `org.etsi.ttcn.tci.TciTypeClass.COMPONENT;`

- `org.etsi.ttcn.tci.TciTypeClass.ENUMERATED;`

- `org.etsi.ttcn.tci.TciTypeClass.FLOAT;`

- `org.etsi.ttcn.tci.TciTypeClass.HEXSTRING;`

- `org.etsi.ttcn.tci.TciTypeClass.INTEGER;`

- `org.etsi.ttcn.tci.TciTypeClass.OBJID;`

- `org.etsi.ttcn.tci.TciTypeClass.OCTETSTRING;`

- `org.etsi.ttcn.tci.TciTypeClass.RECORD;`
- `org.etsi.ttcn.tci.TciTypeClass.RECORD_OF;`
- `org.etsi.ttcn.tci.TciTypeClass.SET;`
- `org.etsi.ttcn.tci.TciTypeClass.SET_OF;`
- `org.etsi.ttcn.tci.TciTypeClass.UNION;`
- `org.etsi.ttcn.tci.TciTypeClass.UNIVERSAL_CHARSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.VERDICT.`

The following constants shall be used for component handling (see also clause 8.2.2.5).

- `org.etsi.ttcn.tci.TciTestComponentKind.TCI_CTRL_COMP;`
- `org.etsi.ttcn.tci.TciTestComponentKind.TCI_MTC_COMP;`
- `org.etsi.ttcn.tci.TciTestComponentKind.TCI_PTC_COMP;`
- `org.etsi.ttcn.tci.TciTestComponentKind.TCI_SYSTEM_COMP;`
- `org.etsi.ttcn.tci.TciTestComponentKind.TCI_ALIVE_COMP.`

8.4 Mapping of interfaces

The TCI IDL definition defines four interfaces, the **TCI-TM,** the **TCI-CH**, the **TCI-CD**, and the **TCI-TL** interface. The operations are defined for different directions within this interface, i.e., some operations can only be called by the TTCN-3 Executable (TE), the System Adaptor (SA) or the Platform Adaptor (PA) on the Test Management and Control (TMC) while others can only be called by the TMC on the TE. This is reflected by dividing the TCI IDL interfaces in two subinterfaces, each suffixed by `Required` or `Provided`.

Table 3 – Subinterfaces

Calling/ Called	TE	TM	CD	CH	SA	PA	TL
TE	–	TCI-TM Provided	TCI-CD Provided	TCI-CH Provided	See [ITU-T Z.165]	See [ITU-T Z.165]	TCI-TL Provided
TM	TCI-TM Required	–	–	–	–	–	TCI-TL Provided
CD	TCI-CD Required	–	–	–	–	–	TCI-TL Provided
CH	TCI-CH Required	–	–	–	–	–	TCI-TL Provided
SA	See [ITU-T Z.165]				–	–	TCI-TL Provided
PA	See [ITU-T Z.165]				–	–	TCI-TL Provided
TL	–	–	–	–	–	–	–

All methods defined in these interfaces should behave as defined in clause 7.

8.4.1 The TCI-TM interface

The **TCI-TM** interface is divided into two subinterfaces, the `TCI-TM Provided` interface, defining calls from the TE to the TM and the `TCI-TM Required` interface, defining calls from the TM to

the TE.

8.4.1.1 TCI-TM provided

The `TCI-TM Provided` interface is mapped to the following interface:

```
// TCI-TM
// TE -> TM
package org.etsi.ttcn.tci;
public interface TciTMProvided {
    public void tciTestCaseStarted (TciTestCaseId testCaseId, TciParameterList
parameterList, Float timer);
    public void     tciTestCaseTerminated ( VerdictValue verdict,
TciParameterList parameterList);
    public void     tciControlTerminated ();
    public Value tciGetModulePar (TciModuleParameterId parameterId);
    public void     tciLog (TriComponentId testComponentId, String message);
    public void     tciError (String message);
}
```

8.4.1.2 TCI-TM required

The `TCI-TM Required` interface is mapped to the following interface:

```
// TCI-TM
// TM -> TE
package org.etsi.ttcn.tci;
public interface TciTMRequired {
    public void                     tciRootModule (TciModuleId moduleName) ;
    public TciModuleParameterList   tciGetModuleParameters (TciModuleId
moduleId);
    public TciTestCaseIdList     tciGetTestCases ();
    public TciParameterTypeList     tciGetTestCaseParameters (TciTestCaseId
TestCaseId);
    public TriPortIdList tciGetTestCaseTSI (TciTestCaseId testCaseId);
    public void             tciStartTestCase(String testCaseId, TciParameterList
parameterList ) ;
    public void             tciStopTestCase () ;
    public TriComponentId tciStartControl () ;
    public void             tciStopControl () ;
}
```

8.4.2 The TCI-CD interface

The **TCI-CD** interface is divided into two subinterfaces, the `TCI-CD Provided` interface, defining calls from the TE to the CD and the `TCI-CD Required` interface, defining calls from the CD to the TE.

8.4.2.1 TCI-CD provided

The `TCI-CD Provided` interface is mapped to the following interface:

```
// TCI-CD
// TE -> CD
package org.etsi.ttcn.tci;
public interface TciCDProvided {
    public Value    decode (TriMessage message, Type decodingHypothesis );
    public TriMessage  encode (Value value);
}
```

8.4.2.2 TCI-CD required

The `TCI-CD Required` interface is mapped to the following interface:

```
// TCI-CD
```

```
// CD -> TE
package org.etsi.ttcn.tci;
public interface TciCDRequired {
    public Type      getTypeForName (String typeName);
    public Type      getInteger ();
    public Type      getFloat ();
    public Type      getBoolean ();
    public Type      getObjid ();
    public Type      getCharstring ();
    public Type      getUniversalCharstring ();
    public Type      getHexstring ();
    public Type      getBitstring ();
    public Type      getOctetstring ();
    public Type      getVerdict ();
    public void      tciErrorReq (String message);
}
```

8.4.3   The TCI-CH interface

The **TCI-CH** interface is divided into two subinterfaces, the `TCI-CH Provided` interface, defining calls from the TE to the CH and the `TCI-CH Required` interface, defining calls from the CH to the TE.

8.4.3.1   TCI-CH provided

The `TCI-CH Provided` interface is mapped to the following interface:

```
// TciCHProvided
// TE -> CH
package org.etsi.ttcn.tci;
public interface TciCHProvided {
    public void tciSendConnected     (TriPortId sender, TriComponentId receiver,
Value sendMessage) ;
    public void tciSendConnectedBC  (TriPortId sender, Value sendMessage) ;
    public void tciSendConnectedMC  (TriPortId sender, TriComponentIdList
receivers, Value sendMessage) ;

    public void       tciCallConnected(TriPortId sender,
                           TriComponentId receiver,
                           TriSignatureId signature,
                           TciParameterList parameterList) ;
    public void       tciCallConnectedBC(TriPortId sender,
                           TriSignatureId signature,
                           TciParameterList parameterList) ;
    public void       tciCallConnectedMC(TriPortId sender,
                           TriComponentIdList receivers,
                           TriSignatureId signature,
                           TciParameterList parameterList) ;

    public void       tciReplyConnected(TriPortId sender,
                           TriComponentId receiver,
                           TriSignatureId signature,
                           TciParameterList parameterList,
                           Value returnValue) ;
    public void       tciReplyConnectedBC(TriPortId sender,
                           TriSignatureId signature,
                           TciParameterList parameterList,
                           Value returnValue) ;
    public void       tciReplyConnectedMC(TriPortId sender,
                           TriComponentIdList receivers,
                           TriSignatureId signature,
                           TciParameterList parameterList,
                           Value returnValue) ;
```

```
    public void        tciRaiseConnected(TriPortId sender,
                            TriComponentId receiver,
                            TriSignatureId signature,
                            Value except) ;
    public void        tciRaiseConnectedBC(TriPortId sender,
                            TriSignatureId signature,
                            Value except) ;
    public void        tciRaiseConnectedMC(TriPortId sender,
                            TriComponentIdList receivers,
                            TriSignatureId signature,
                            Value except) ;

    public TriComponentId tciCreateTestComponentReq(int kind, Type
componentType, String name) ;

    public void        tciStartTestComponentReq(TriComponentId comp,
                                TciBehaviourId behavior,
                                TciParameterList parameterList) ;

    public void        tciStopTestComponentReq(TriComponentId comp) ;

    public void        tciConnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void        tciDisconnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void        tciTestComponentTerminatedReq(TriComponentId comp,
VerdictValue verdict) ;

    public boolean  tciTestComponentRunningReq(TriComponentId comp) ;

    public TriComponentId tciGetMTCReq() ;

    public void        tciMapReq(TriPortId fromPort, TriPortId toPort);

    public void        tciUnmapReq(TriPortId fromPort, TriPortId toPort);

    public void        tciExecuteTestCaseReq(TriComponentId testComponentId,
                            TriPortIdList tsiPortList);

    public void        tciResetReq() ;

    public boolean  tciTestComponentDoneReq(TriComponentId testComponentId) ;

    public void  tciKillTestComponentReq(TriComponentId component)

    public boolean  tciTestComponentAliveReq (TriComponentId component)

    public boolean  tciTestComponentKilledReq (TriComponentId component)
}
```

8.4.3.2   TCI-CH required

The `TCI-CH Required` interface is mapped to the following interface:

```
// TciCHRequired
// CH -> TE
package org.etsi.ttcn.tci;
public interface TciCHRequired extends TciCDRequired {
    public void        tciEnqueueMsgConnected(TriPortId sender,
                                TriComponentId receiver,
                                Value receivedMessage) ;

    public void        tciEnqueueCallConnected(TriPortId sender,
                                TriComponentId receiver,
```

```
                                  TriSignatureId signature,
                                  TciParameterList parameterList) ;

    public void        tciEnqueueReplyConnected(TriPortId sender,
                                  TriComponentId receiver,
                                  TriSignatureId signature,
                                  TciParameterList parameterList,
                                  Value returnValue) ;

    public void        tciEnqueueRaiseConnected(TriPortId sender,
                                  TriComponentId receiver,
                                  TriSignatureId signature,
                                  Value except) ;

    public TriComponentId tciCreateTestComponent(int kind, Type componentType,
String name) ;

    public void        tciStartTestComponent(TriComponentId comp,
                                  TciBehaviourId behavior,
                                  TciParameterList parameterList) ;

    public void        tciStopTestComponent(TriComponentId comp) ;

    public void        tciConnect(TriPortId fromPort, TriPortId toPort) ;

    public void        tciDisconnect(TriPortId fromPort, TriPortId toPort) ;

    public void        tciTestComponentTerminated(TriComponentId comp, VerdictValue
verdict);

    public boolean  tciTestComponentRunning(TriComponentId comp);

    public boolean  tciTestComponentDone(TriComponentId comp);

    public TriComponentId tciGetMTC ();

    public void        tciExecuteTestCase (TciTestCaseId TestCaseId, TriPortIdList
tsiPortList);

    public void        tciReset ();

    public void        tciMap (TriPortId fromPort, TriPortId toPort);

    public void        tciUnmap (TriPortId fromPort, TriPortId toPort);

    public void        tciKillTestComponent(TriComponentId component)

    public boolean  tciTestComponentAlive (TriComponentId component)

    public boolean  tciTestComponentKilled (TriComponentId component)
}
```

8.4.4   The TCI-TL interface

The **TCI-TL** interface contains only one subinterface, the TCI-TL Provided interface, defining calls from the TE, TM, CH, CD, SA and PA to the TL.

8.4.4.1   TCI-TL provided

The TCI-TL Provided interface is mapped to the following interface:

```
// TCI-TL
// TE,TM,CH,CD,SA,PA -> TL
package org.etsi.ttcn.tci;
```

```java
public interface TciTLProvided {
    public void tliTcExecute(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TriParameterList triPars, TriTimerDuration dur);
    public void tliTcStart(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TciParameterList tciPars, TriTimerDuration dur);
    public void tliTcStop(String am, int ts, String src, int line, TriComponentId c);
    public void tliTcStarted(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TciParameterList tciPars, TriTimerDuration dur);
    public void tliTcTerminated(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TciParameterList tciPars, VerdictValue verdict);
    public void tliCtrlStart(String am, int ts, String src, int line, TriComponentId c);
    public void tliCtrlStop(String am, int ts, String src, int line, TriComponentId c);
    public void tliCtrlTerminated(String am, int ts, String src, int line, TriComponentId c);
    public void tliMSend_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, Value msgValue, TriAddress address,
            TciStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, Value msgValue
            TciStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, Value msgValue, TriAddressList addresses,
            TciStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, Value msgValue, TriStatus transmissionFailure);
    public void tliMSend_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, Value msgValue, TriStatus transmissionFailure);
    public void tliMSend_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, Value msgValue, TriStatus transmissionFailure);
    public void tliMDetected_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriMessage msg, TriAddress address);
    public void tliMDetected_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, Value msgValue );
    public void tliMMismatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliMMismatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliMReceive_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, Value msgValue, TciValueTemplate msgTmpl,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliMReceive_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, Value msgValue, TciValueTemplate msgTmpl,
            TriComponentId fromComp, TciNonValueTemplate fromTmpl);
    public void tliPrCall_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            TriAddress address, TciStatus encoderFailure, TriParameterList triPars,
            TriStatus transmissionFailure);
    public void tliPrCall_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            TciStatus encoderFailure, TriParameterList triPars,
            TriStatus transmissionFailure);
    public void tliPrCall_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            TriAddressList addresses, TciStatus encoderFailure, TriParameterList triPars,
            TriStatus transmissionFailure);
    public void tliPrCall_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            TriStatus transmissionFailure);
    public void tliPrCall_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            TriStatus transmissionFailure);
    public void tliPrCall_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            TriStatus transmissionFailure);
    public void tliPrGetCallDetected_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriSignatureId signature, TriParameterList triPars,
TriAddress address);
    public void tliPrGetCallDetected_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriSignatureId signature, TciParameterList tciPars );
    public void tliPrGetCallMismatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrGetCallMismatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrGetCall_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
```

```
            TciParameterList tciPars, TciValueTemplate parsTmpl,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrGetCall_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, TciValueTemplate parsTmpl,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrReply_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            Value replValue, TriAddress address,
            TciStatus encoderFailure, TriParameterList triPars,
            TriParameter repl, TriStatus transmissionFailure);
    public void tliPrReply_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            Value replValue, TciStatus encoderFailure,
            TriParameterList triPars, TriParameter repl, TriStatus transmissionFailure);
    public void tliPrReply_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
            Value replValue, TriAddressList addresses,
            TciStatus encoderFailure, TriParameterList triPars,
            TriParameter repl, TriStatus transmissionFailure);
    public void tliPrReply_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature,
            TciParameterList tciPars, Value replValue,
            TriStatus transmissionFailure);
    public void tliPrReply_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            Value replValue, TriStatus transmissionFailure);
    public void tliPrReply_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            Value replValue, TriStatus transmissionFailure);
    public void tliPrGetReplyDetected_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriSignatureId signature, in TriParameterListType triPars,
            TriParameter repl, TriAddress address);
    public void tliPrGetReplyDetected_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriSignatureId signature, in TciParameterList tciPars,
            Value replValue);
    public void tliPrGetReplyMismatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, in TciValueTemplate parsTmpl,
            Value replValue, TciValueTemplate replyTmpl, TciValueDifferenceList diffs,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrGetReplyMismatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, in TciValueTemplate parsTmpl,
            Value replValue, TciValueTemplate replyTmpl, TciValueDifferenceList diffs,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrGetReply_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, in TciValueTemplate parsTmpl,
            Value replValue, TciValueTemplate replyTmpl,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrGetReply_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            TciParameterList tciPars, in TciValueTemplate parsTmpl,
            Value replValue, TciValueTemplate replyTmpl,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrRaise_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to,
            TriSignatureId signature, TciParameterList tciPars, Value excValue,
            TriAddress address, TciStatus encoderFailure, TriException exc,
            TriStatus transmissionFailure);
    public void tliPrRaise_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to,
            TriSignatureId signature, TciParameterList tciPars, Value excValue,
            TciStatus encoderFailure, TriException exc, TriStatus transmissionFailure);
    public void tliPrRaise_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to,
            TriSignatureId signature, TciParameterList tciPars, Value excValue,
            TriAddressList addresses, TciStatus encoderFailure, TriException exc,
            TriStatus transmissionFailure);
    public void tliPrRaise_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId to, TriSignatureId signature,
            TciParameterList tciPars, Value excValue, TriStatus transmissionFailure);
    public void tliPrRaise_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            Value excValue, TriStatus transmissionFailure);
    public void tliPrRaise_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
            Value excValue, TriStatus transmissionFailure);
    public void tliPrCatchDetected_m(String am, int ts, String src, int line, TriComponentId c,
```

```
            TriPortId at, TriPortId from, TriSignatureId signature,
            TriException exc, TriAddress address);
    public void tliPrCatchDetected_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriPortId from, TriSignatureId signature,
            Value excValue);
    public void tliPrCatchMismatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            Value excValue, TciValueTemplate excTmpl,  TciValueDifferenceList diffs,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrCatchMismatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            Value excValue, TciValueTemplate excTmpl,  TciValueDifferenceList diffs,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrCatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            Value excValue, TciValueTemplate excTmpl,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrCatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature,
            Value excValue, TciValueTemplate excTmpl,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrCatchTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature);
    public void tliPrCatchTimeout(String am, int ts, String src, int line, TriComponentId c,
            TriPortId at, TriSignatureId signature);
    public void tliCCreate(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, String name, Boolean alive);
    public void tliCStart(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, TciBehaviourId name, TciParameterList tciPars);
    public void tliCRunning(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, ComponentStatus status);
    public void tliCAlive(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, ComponentStatus status);
    public void tliCStop(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp);
    public void tliCKill(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp);
    public void tliCDoneMismatch(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, TciNonValueTemplate compTmpl);
    public void tliCDone(String am, int ts, String src, int line, TriComponentId c,
            TciNonValueTemplate compTmpl);
    public void tliCKilledMismatch(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId comp, TciNonValueTemplate compTmpl);
    public void tliCKilled(String am, int ts, String src, int line, TriComponentId c,
            TciNonValueTemplate compTmpl);
    public void tliCTerminated(String am, int ts, String src, int line, TriComponentId c,
            VerdictValue verdict);
    public void tliPConnect(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port1, TriPortId port2);
    public void tliPDisconnect(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPMap(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port1, TriPortId port2);
    public void tliPUnmap(String am, int ts, String src, int line, TriComponentId c,
            TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPClear(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port);
    public void tliPStart(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port);
    public void tliPStop(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port);
    public void tliPHalt(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port);
    public void tliEncode(String am, int ts, String src, int line, TriComponentId c,
            Value val, TciStatus encoderFailure, TriMessage msg, String codec);
    public void tliDecode(String am, int ts, String src, int line, TriComponentId c,
            TriMessage msg, TciStatus decoderFailure, Value val, String codec);
    public void tliTTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer);
    public void tliTTimeoutMismatch(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, TciNonValueTemplate timerTmpl);
    public void tliTTimeout(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, TciNonValueTemplate timerTmpl);
    public void tliTStart(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, TriTimerDuration dur);
    public void tliTStop(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, in TriTimerDuration dur);
    public void tliTRead(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, TriTimerDuration elapsed);
    public void tliTRunning(String am, int ts, String src, int line, TriComponentId c,
```

```
                TriTimerId timer, TimerStatus status);
        public void tliSEnter(String am, int ts, String src, int line, TriComponentId c,
                QualifiedName name, TciParameterList tciPars, String kind);
        public void tliSLeave(String am, int ts, String src, int line, TriComponentId c,
                QualifiedName name, TciParameterListType tciPars, Value returnValue, String kind);
        public void tliVar(String am, int ts, String src, int line, TriComponentId c,
                QualifiedName name, Value varValue);
        public void tliModulePar(String am, int ts, String src, int line, TriComponentId c,
                QualifiedName name, Value parValue);
        public void tliGetVerdict(String am, int ts, String src, int line, TriComponentId c,
                VerdictValue verdict);
        public void tliSetVerdict(String am, int ts, String src, int line, TriComponentId c,
                VerdictValue verdict);
        public void tliLog(String am, int ts, String src, int line, TriComponentId c,
                TString log);
        public void tliAEnter(String am, int ts, String src, int line, TriComponentId c);
        public void tliALeave(String am, int ts, String src, int line, TriComponentId c);
        public void tliADefaults(String am, int ts, String src, int line, TriComponentId c);
        public void tliAActivate(String am, int ts, String src, int line, TriComponentId c,
                QualifiedName name, TciParameterList tciPars, Value ref);
        public void tliADeactivate(String am, int ts, String src, int line, TriComponentId c,
                Value ref);
        public void tliANomatch(String am, int ts, String src, int line, TriComponentId c);
        public void tliARepeat(String am, int ts, String src, int line, TriComponentId c);
        public void tliAWait(String am, int ts, String src, int line, TriComponentId c);
        public void tliAction(String am, int ts, String src, int line, TriComponentId c, String action);
        public void tliMatch(String am, int ts, String src, int line, TriComponentId c, Value expr,
                TciValueTemplate tmpl);
        public void tliMatchMismatch(String am, int ts, String src, int line, TriComponentId c,
                Value expr, TciValueTemplate tmpl, TciValueDifferenceList diffs);
        public void tliInfo (String am, int ts, String src, int line, TriComponent c,
                int level, String info)
}
```

8.5   Optional parameters

Clause 7.3 defines that a reserved value shall be used to indicate the absence of an optional parameter. For the Java language mapping the Java `null` value shall be used to indicate the absence of an optional value. For example, if in the `tciReplyConnected` operation the value parameter shall be omitted the operation invocation shall be `tciReplyConnected (sender, receiver, signature, parameterList, null)`.

8.6   TCI initialization

All methods are non-static, i.e., operations can only be called on objects. As this Recommendation does not define concrete implementation strategies of TE, TM, CD and CH the mechanism how the TE, the TM, the CD or the CH get to know the handles on the respective objects is out of scope of this Recommendation.

Tool vendors shall provide methods to the developers of TM, CD and CH to register the TE, TM, CD and CH to their respective communication partner.

8.7   Error handling

All operations called from the TM, CH or CD that return have succeeded. If an erroneous situation has been identified by the TE a test case error will be communicated to the user using the procedures as defined in clause 7.3.1.2.5 (`tciError`). If an operation called by the TE in the TM, CH, CD, or TL produces an error, this erroneous situation should be communicated to the TE using the procedures as defined in clause 7.3.2.1.12 (`tciErrorReq`).

Beside this error handling, no additional error handling is defined with this Java language mapping. In particular, no exception handling mechanisms are defined.

9   ANSI C language mapping

9.1   Introduction

This clause defines the TRI ANSI-C language mapping for the TCI data specified in clause 7.2 and for the TCI operations specified in clause 7.3.

9.2 Value interfaces

TCI IDL interface	ANSI C representation	Notes and comments
Type		
TciModuleIdType getDefiningModule()	`TciModuleIdType tciGetDefiningModule(TciType inst)`	
TString getName()	`String tciGetName(TciType inst)`	String type reused from IDL (OMG recommendation)
TciTypeClassType getTypeClass()	`TciTypeClassType tciGetTypeClass (TciType inst)`	
Value newInstance()	`TciValue tciNewInstance(TciType inst)`	
TString getTypeEncoding()	`String tciGetTypeEncoding(TciType inst)`	
TStringSeq getTypeExtension()	`String* getTypeExtension(TciType inst)`	
TString getTypeEncodingVariant()	`String tciGetTypeEncodingVariant(TciType inst)`	
Value		
TString getValueEncoding()	`String tciGetValueEncoding(TciValue inst)`	
TString getValueEncodingVariant()	`String tciGetValueEncodingVariant(TciValue inst)`	
Type getType()	`TciType tciGetType(TciValue inst)`	
TBoolean notPresent()	`Boolean tciNotPresent(TciValue inst)`	Boolean type reused from IDL (OMG recommendation)
IntegerValue		
TInteger getInt()	`String tciGetIntAbs(TciValue inst)`	Returns the (10-base) integer absolute value as an ASCII string.
	`unsigned long int tciGetIntNumberOfDigits (TciValue inst)`	Returns the number of digits in an integer value.
	`Boolean tciGetIntSign(TciValue inst)`	Returns true if the number is positive, false otherwise.

TCI IDL interface	ANSI C representation	Notes and comments
	``char tciGetIntDigit
(TciValue inst,
unsigned long int position)```	Returns the value of the digit at position 'position', where position 0 is the least significant digit.
void setInt(in TInteger value)	```void tciSetIntAbs(TciValue inst, String value)```	Sets the (10-base) absolute value of the integer via an ASCII string. The first digit must not be 0 unless the value is 0.
	``void tciSetIntNumberOfDigits
(TciValue inst,
unsigned long int dig_num)```	Sets the number of digits in an integer value.
	``void tciSetIntSign
(TciValue inst,
Boolean sign)```	Sets the sign to + (true) or – (false).
	``void tciSetIntDigit
(TciValue inst,
unsigned long int position,
char digit)```	Sets the value of the digit at position 'position', where position 0 is the least significant digit.
FloatValue		
TFloat getFloat()	```double tciGetFloatValue(TciValue inst)```	
void setFloat(in TFloat value)	```void tciSetFloatValue(TciValue inst, double value)```	
BooleanValue		
TBoolean getBoolean()	```Boolean tciGetBooleanValue(TciValue inst)```	
void setBoolean (in TBoolean value)	```void tciSetBooleanValue
(TciValue inst, Boolean value)```	
ObjidValue		
TObjid getObjid()	```TciObjidValue tciGetTciObjidValue(TciValue inst)```	
void setObjid(in TObjid value)	```void tciSetObjidValue(TciValue inst,
TciObjidValue value)```	
CharstringValue		
TString getString()	```TciCharStringValue
tciGetCStringValue(TciValue inst)```	
void setString(in TString value)	```void tciSetCStringValue
(TciValue inst, TciCharString value)```	
TChar getChar (in TInteger position)	```char tciGetCStringCharValue
(TciValue inst, long int position)```	

TCI IDL interface	ANSI C representation	Notes and comments

void setChar
 	
 (in TInteger position, in char value)
	```void tciSetCStringCharValue```
	``` (TciValue inst,```
	``` long int position,```
	``` char value)```	

TInteger getLength()	```unsigned long int tciGetCStringLength``` ```(TciValue inst)```	

void setLength(in TInteger len)	```void tciSetCStringLength```
	``` (TciValue inst,```
	``` unsigned long int len)```	

UniversalCharstringValue		
TString getString()	```TciUCStringValuetciGetUCStringValue(TciValue``` ```inst)```	
void setString(in TString value)	```void tciSetUCStringValue``` ``` (TciValue inst, TciUCStringValue value)```	

TUniversalChar getChar
 	
 (in TInteger position)
	```void tciGetUCStringCharValue```
	``` (TciValue inst, unsigned long int position,```
	``` TciUCValue result)```	

void setChar
 	
 (in TInteger position,
 	
 in TUniversalChar value)
	```void tciSetUCStringCharValue```
	``` (TciValue inst,```
	``` unsigned long int position,```
	``` TciUCValue value)```	

TInteger getLength()	```unsigned long int``` ```tciGetUCStringLength(TciValue inst)```	

void setLength(in TInteger len)	```void tciSetUCStringLength```
	``` (TciValue inst,```
	``` unsigned long int len)```	

BitstringValue		
TString getString()	```String tciGetBStringValue(TciValue inst)```	
void setString(in TString value)	```void tciSetBStringValue``` ``` (TciValue inst, String value)```	
TChar getBit (in integer position)	```int tciGetBStringBitValue``` ``` (TciValue inst, long int position)```	

void setBit
 	
 (in TInteger position,
 	
 in TInteger value)
	```void tciSetBStringBitValue```
	``` (TciValue inst,```
	``` unsigned long int position,```
	``` int value)```	

TInteger getLength()	```unsigned long int``` ```tciGetBStringLength(TciValue inst)```	

void setLength(in TInteger len)	```void tciSetBStringLength```
	``` (TciValue inst,```
	``` long int len)```	

TCI IDL interface	ANSI C representation	Notes and comments
HexstringValue		
TString getString()	`String tciGetHStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetHStringValue` `  (TciValue inst, String value)`	
TChar getHex (in TInteger position)	`int tciGetHStringHexValue` `  (TciValue inst, unsigned long int position)`	

void setBit
 (in TInteger position,
 in TInteger value)
	`void tciSetHStringHexValue`
	`  (TciValue inst,`
	`   unsigned long int position,`
	`   int value)`	

TInteger getLength()	`long int tciGetHStringLength(TciValue inst)`	

void setLength(in TInteger len)	`void tciSetHStringLength`
	`  (TciValue inst,`
	`   unsigned long int len)`	

OctetstringValue		
TString getString()	`String tciGetOStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetOStringValue` `  (TciValue inst, String value)`	
TChar getOctet(in TInteger position)	`int tciGetOStringOctetValue` `  (TciValue inst, unsigned long int position)`	

void setOctet
 (in TInteger position,
 in TInteger value)
	`void tciSetOStringOctetValue`
	`  (TciValue inst,`
	`   unsigned long int position,`
	`   int value)`	

TInteger getLength()	`unsigned long int` `tciGetOStringLength(TciValue inst)`	

void setLength(in TInteger len)	`void tciSetOStringLength`
	`  (TciValue inst,`
	`   unsigned long int len)`	

RecordValue		

Value getField(in TString fieldName)	`TciValue tciGetRecFieldValue`
	`  (TciValue inst,`
	`   String fieldName)`	

void setField
 (in TString fieldName,
 in Value value)
	`void tciSetRecFieldValue`
	`  (TciValue inst,`
	`   String fieldName,`
	`   TciValue value)`	

TString[] getFieldNames()	`char** tciGetRecFieldNames(TciValue inst)`	Returns a NULL-terminated array of the field names.
void setFieldOmitted (in TString fieldName)	`void setFieldOmitted` `  (String fieldName)`	

TCI IDL interface	ANSI C representation	Notes and comments
RecordOfValue		

Value getField(in TInteger position)	`TciValue tciGetRecOfFieldValue`
	`  (TciValue inst,`
	`   unsigned long int position)`	

void setField
 (in TInteger position,
 in Value value)
	`void tciSetRecOfFieldValue`
	`  (TciValue inst,`
	`   unsigned long int position,`
	`   TciValue value)`	

void appendField(in Value value)	`void tciAppendRecOfFieldValue`
	`  (TciValue inst,`
	`   TciValue value)`	

Type getElementType()	`TciType tciGetRecOfElementType(TciValue inst)`	
TInteger getLength()	`unsigned long int tciGetRecOfLength(TciValue inst)`	

void setLength(in TInteger len)	`void tciSetRecOfLength`
	`  (TciValue inst,`
	`   unsigned long int len)`	

UnionValue		

Value getVariant
 (in TString variantName)
	`TciValue tciGetUnionVariant`
	`  (TciValue inst,`
	`   String variantName)`	

void setVariant
 (in TString variantName,
 in Value value)
	`void tciSetUnionVariant`
	`  (TciValue inst,`
	`   String variantName,`
	`   TciValue value)`	

TString getPresentVariantName()	`String tciGetUnionPresentVariantName` `  (TciValue inst)`	
TString[] getVariantNames()	`char** tciGetUnionVariantNames(TciValue inst)`	Returns a NULL-terminated array of the field names.
EnumeratedValue		
TString getEnum()	`String tciGetEnumValue(TciValue inst)`	

void setEnum(in TString enumValue)	`void tciSetEnumValue`
	`  (TciValue inst,`
	`   String enumValue)`	

VerdictValue		
TInteger getVerdict()	`int tciGetVerdictValue(TciValue inst)`	
void setVerdict(in TInteger verdict)	`void tciSetVerdictValue(TciValue inst, int verdict)`	

TCI IDL interface	ANSI C representation	Notes and comments
AddressValue		
Value getAddress()	`TciValue tciGetAddressValue(TciValue inst)`	
void setAddress(in Value value)	`void tciSetAddressValue(TciValue inst, TciValue value)`	

9.3 Logging interface

TCI IDL Interface	ANSI C representation	Notes and comments
TciValueTemplate		
TBoolean isOmit()	`Boolean tciIsOmit(TciValueTemplate inst)`	Boolean type reused from IDL (OMG recommendation)
TBoolean isAny()	`Boolean tciIsAny(TciValueTemplate inst)`	Boolean type reused from IDL (OMG recommendation)
TBoolean isAnyOrOmit()	`Boolean tciIsAnyOrOmit(TciValueTemplate inst)`	Boolean type reused from IDL (OMG recommendation)
TString getTemplateDef()	`String tciGetTemplateDef(TciValueTemplate inst)`	String type reused from IDL (OMG recommendation)
TciNonValueTemplate		
TBoolean isAny()	`Boolean tciIsAny(TciNonValueTemplate inst)`	Boolean type reused from IDL (OMG recommendation)
TBoolean isAll()	`Boolean tciIsAll(TciNonValueTemplate inst)`	Boolean type reused from IDL (OMG recommendation)
TString getTemplateDef()	`String tciGetTemplateDef(TciNonValueTemplate inst)`	String type reused from IDL (OMG recommendation)

9.4 Operation interfaces

TCI-TM required		
void tciRootModule (in TciModuleIdType moduleId)	`void tciRootModule(String moduleId)`	
TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName)	`TciModuleParameterListType tciGetModuleParameters (TciModuleIdType moduleName)`	
TciTestCaseIdListType tciGetTestCases()	`TciTestCaseIdListType tciGetTestCases()`	

TciParameterTypeListType tciGetTestCaseParameters (in TciTestCaseIdType testCaseId)	`TciParameterTypeListType`
	`  tciGetTestCaseParameters`
	`    (TciTestCaseIdType`
	`testCaseId)`	

TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)	`TriPortIdList tciGetTestCaseTSI` `   (TciTestCaseIdType testCaseId)`	

void tciStartTestCase (in TciTestCaseIdType testCaseId, in TciParameterListType parameterlist)	`void tciStartTestCase`
	`  (TciTestCaseIdType testCaseId,`
	`   TciParameterListType`
	`parameterlist)`	

void tciStopTestCase()	`void tciStopTestCase()`	
TriComponentId tciStartControl()	`TriComponentId tciStartControl()`	
void tciStopControl()	`void tciStopControl()`	
TCI-TM provided		

void tciTestCaseStarted (in TciTestCaseIdType testCaseId, in TciParameterListType parameterList, in Tfloat timer)	`void tciTestCaseStarted`
	`  (TciTestCaseIdType testCaseId,`
	`   TciParameterListType`
	`parameterList,`
	`    double timer)`	

void tciTestCaseTerminated (in VerdictValue verdict, in TciParameterListType parameterlist)	`void tciTestCaseTerminated`
	`  (TciVerdictValue verdict,`
	`   TciParameterListType`
	`parameterlist)`	

void tciControlTerminated()	`void tciControlTerminated()`	

Value tciGetModulePar (in TciModuleParameterIdType parameterId)	`tciValue tciGetModulePar`
	`  (TciModuleParameterIdType`
	`parameterId)`	

void tciLog(in TString message)	`void tciLog(String message)`	
void tciError(in TString message)	`void tciError(String message)`	
TCI-CD required		
Type getTypeForName (in String typeName)	`TciType tciGetTypeForName` `  (String typeName)`	
Type getInteger()	`TciType tciGetIntegerType()`	
Type getFloat()	`TciType tciGetFloatType()`	
Type getBoolean()	`TciType tciGetBooleanType()`	
Type getChar()	`TciType tciGetCharType()`	
Type getUniversalChar()	`TciType` `tciGetUniversalCharType()`	
Type getObjid()	`TciType tciGetTciObjidType()`	
Type getCharstring()	`TciType` `tciGetTciCharstringType()`	
Type getUniversalCharstring()	`TciType` `tciGetUniversalCharstringType()`	
Type getHexstring()	`TciType tciGetHexstringType()`	
Type getBitstring()	`TciType tciGetBitstringType()`	

Type getOctetstring()	`TciType tciGetOctetstringType()`	
Type getVerdict()	`TciType tciGetVerdictType()`	
void tciErrorReq(in String message)	`void tciErrorReq(String message)`	
TCI-CD provided		

Value decode
 (in TriMessageType message,
 in Type decodingHypothesis)
	`TciValue tciDecode`
	`  (BinaryString message,`
	`   TciType decHypothesis)`
		BinaryString type reused from TRI

TriMessageType encode (in Value value)	`BinaryString tciEncode(TciValue value)`	
TCI-CH required		

void tciEnqueueMsgConnected
 (in TriPortIdType sender,
 in TriComponentIdType receiver,
 in Value rcvdMessage)
	`void tciEnqueueMsgConnected`
	`  (TriPortId sender,`
	`   TriComponentId receiver,`
	`   TciValue rcvdMessage)`	

void tciEnqueueCallConnected
 (in TriPortIdType sender,
 in TriComponentIdType receiver,
 in TriSignatureIdType signature,
 in TciParameterListType parameterList)
	`void tciEnqueueCallConnected`
	`  (TriPortId sender,`
	`   TriComponentId receiver,`
	`   TriSignatureId signature,`
	`   TciParameterListType parameterList)`	

void tciEnqueueReplyConnected
 (in TriPortIdType sender,
 in TriComponentIdType receiver,
 in TriSignatureIdType signature,
 in TciParameterListType parameterList,
 in Value returnValue)
	`void tciEnqueueReplyConnected`
	`  (TriPortId sender,`
	`   TriComponentId receiver,`
	`   TriSignatureId signature,`
	`   TciParameterListType parameterList,`
	`   TciValue returnValue)`	

void tciEnqueueRaiseConnected
 (in TriPortIdType sender,
 in TriComponentIdType receiver,
 in TriSignatureIdType signature,
 in Value exception)
	`void tciEnqueueRaiseConnected`
	`  (TriPortId sender,`
	`   TriComponentId receiver,`
	`   TriSignatureIdType signature,`
	`   TciValue exception)`	

TriComponentIdType tciCreateTestComponent
 (in TciTestComponentKindType kind
 in Type componentType, in TString name)
	`TriComponentId tciCreateTestComponent`
	`  (TciTestComponentKindType kind,`
	`   TciType componentType,`
	`   String name)`	

void tciStartTestComponent
 (in TriComponentIdType component,
 in TciBehaviourIdType behavior,
 in TciParamaterListType parameterList)
	`void tciStartTestComponent`
	`  (TriComponentId component,`
	`   TciBehaviourIdType behavior,`
	`   TciParamaterListType parameterList)`	

void tciStopTestComponent (in TriComponentIdType component)	`void tciStopTestComponent` `  (TriComponentId component)`	

void tciConnect
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciConnect`
	`  (TriPortId fromPort,`
	`   TriPortId toPort)`	

void tciDisconnect
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciDisconnect`
	`  (TriPortId fromPort,`
	`   TriPortId toPort)`	

void tciMap
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciMap`
	`   (TriPortId fromPort,`
	`    TriPortId toPort)`	

void tciUnmap
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciUnmap`
	`   (TriPortId fromPort,`
	`    TriPortId toPort)`	

void tciTestComponentTerminated
 (in TriComponentIdType component,
 in VerdictValue verdict)
	`void tciTestComponentTerminated`
	`  (TriComponentId component,`
	`   TciVerdictValue verdict)`	

boolean tciTestComponentRunning (in TriComponentIdType component)	`Boolean tciTestComponentRunning` `  (TriComponentId component)`	
boolean tciTestComponentDone (in TriComponentIdType component)	`boolean tciTestComponentDone` `   (TriComponentId component)`	
TriComponentIdType tciGetMTC()	`TriComponentId tciGetMTC()`	

void tciExecuteTestCase
 (in TciTestCaseIdType testCaseId,
 in TriPortIdListType tsiPortList)
	`void tciExecuteTestCase`
	`  (TciTestCaseIdType testCaseId,`
	`   TriPortIdList tsiPortList)`	

void tciReset()	`void tciReset()`	
void tciKillTestComponent (in TriComponentIdType component)	`void tciKillTestComponent` `  (TriComponentId component)`	
TBoolean tciTestComponentAlive (in TriComponentIdType component)	`Boolean tciTestComponentAlive` `  (TriComponentId component)`	
TBoolean tciTestComponentKilled (in TriComponentIdType component)	`Boolean tciTestComponentKilled` `  (TriComponentId component)`	
TCI-CH provided		

void tciSendConnected
 (in TriPortIdType sender,
 in TriComponentIdType receiver,
 in Value sendMessage)
	`void tciSendConnected`
	`  (TriPortId sender,`
	`   TriComponentId receiver,`
	`   TciValue sendMessage)`	

void tciSendConnectedBC
 (in TriPortIdType sender,
 in Value sendMessage)
	`void tciSendConnectedBC`
	`  (TriPortId sender,`
	`   TciValue sendMessage)`	

void tciSendConnectedMC
 (in TriPortIdType sender,
 in TriComponentIdListType receivers,
 in Value sendMessage)
	`void tciSendConnectedMC`
	`  (TriPortId sender,`
	`   TriComponentIdList receivers,`
	`   TciValue sendMessage)`	

| void tciCallConnected<br>(in TriPortIdType sender,<br>in TriComponentIdType receiver,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList) | ```
void tciCallConnected
  (TriPortId sender,
   TriComponentId receiver,
   TriSignatureId signature,
   TciParameterListType
parameterList)
``` | |
| void tciCallConnectedBC<br>(in TriPortIdType sender,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList) | ```
void tciCallConnectedBC
  (TriPortId sender,
   TriSignatureId signature,
   TciParameterListType
parameterList)
``` | |
| void tciCallConnectedMC<br>(in TriPortIdType sender,<br>in TriComponentIdListType receivers,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList) | ```
void tciCallConnectedMC
  (TriPortId sender,
   TriComponentIdList receivers,
   TriSignatureId signature,
   TciParameterListType
parameterList)
``` | |
| void tciReplyConnected<br>(in TriPortIdType sender,<br>in TriComponentIdType receiver,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList,<br>in Value returnValue) | ```
void tciReplyConnected
  (TriPortId sender,
   TriComponentId receiver,
   TriSignatureIdType signature,
   TciParameterListType
parameterList,
   TciValue returnValue)
``` | |
| void tciReplyConnectedBC<br>(in TriPortIdType sender,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList,<br>in Value returnValue) | ```
void tciReplyConnectedBC
  (TriPortId sender,
   TriSignatureIdType signature,
   TciParameterListType
parameterList,
   TciValue returnValue)
``` | |
| void tciReplyConnectedMC<br>(in TriPortIdType sender,<br>in TriComponentIdListType receivers,<br>in TriSignatureIdType signature,<br>in TciParameterListType parameterList,<br>in Value returnValue) | ```
void tciReplyConnectedMC
  (TriPortId sender,
   TriComponentIdList receivers,
   TriSignatureIdType signature,
   TciParameterListType
parameterList,
   TciValue returnValue)
``` | |
| void tciRaiseConnected<br>(in TriPortIdType sender,<br>in TriComponentIdType receiver,<br>in TriSignatureIdType signature,<br>in Value exception) | ```
void tciRaiseConnected
  (TriPortId sender,
   TriComponentId receiver,
   TriSignatureId signature,
   TciValue exception)
``` | |
| void tciRaiseConnectedBC<br>(in TriPortIdType sender,<br>in TriSignatureIdType signature,<br>in Value exception) | ```
void tciRaiseConnectedBC
  (TriPortId sender,
   TriSignatureId signature,
   TciValue exception)
``` | |

void tciRaiseConnectedMC
 (in TriPortIdType sender,
 in TriComponentIdListType receivers,
 in TriSignatureIdType signature,
 in Value exception)
	`void tciRaiseConnectedMC`
	`  (TriPortId sender,`
	`   TriComponentIdList receivers,`
	`   TriSignatureId signature,`
	`   TciValue exception)`	

TriComponentIdType
tciCreateTestComponentReq
 (in TciTestComponentKindType kind,
 in Type componentType, in TString name)
	`TriComponentId`
	`tciCreateTestComponentReq`
	`  (TciTestComponentKindType kind,`
	`   TciType componentType,`
	`   String name)`	

void tciStartTestComponentReq
 (in TriComponentIdType component,
 in TciBehaviourIdType behavior,
 in TciParamaterListType parameterList)
	`void tciStartTestComponentReq`
	`  (TriComponentId component,`
	`   TciBehaviourIdType behavior,`
	`   TciParamaterListType`
	`parameterList)`	

void tciStopTestComponentReq (in TriComponentIdType component)	`void tciStopTestComponentReq` `  (TriComponentId component)`	

void tciConnectReq
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciConnectReq`
	`  (TriPortId fromPort,`
	`   TriPortId toPort)`	

void tciDisconnectReq
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciDisconnectReq`
	`  (TriPortId fromPort,`
	`   TriPortId toPort)`	

void tciMapReq
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciMapReq`
	`   (TriPortId fromPort,`
	`    TriPortId toPort)`	

void tciUnmapReq
 (in TriPortIdType fromPort,
 in TriPortIdType toPort)
	`void tciUnmapReq`
	`   (TriPortId fromPort,`
	`    TriPortId toPort)`	

void tciTestComponentTerminatedReq
 (in TriComponentIdType component,
 in VerdictValue verdict)
	`void`
	`tciTestComponentTerminatedReq`
	`  (TriComponentId component,`
	`   TciVerdictValue verdict)`	

boolean tciTestComponentRunningReq
 (in TriComponentIdType component)
	`Boolean`
	`tciTestComponentRunningReq`
	`  (TriComponentId component)`	

boolean tciTestComponentDoneReq (in TriComponentIdType component)	`Boolean tciTestComponentDoneReq` `  (TriComponentId component)`	
TriComponentIdType tciGetMTCReq()	`TriComponentId tciGetMTCReq()`	

void tciExecuteTestCaseReq
 (in TciTestCaseIdType testCaseId,
 in TriPortIdListType tsiPortList)
	`void tciExecuteTestCaseReq`
	`   (TciTestCaseIdType testCaseId,`
	`    TriPortIdList tsiPortList)`	

void tciResetReq()	`void tciResetReq()`	
void tciKillTestComponentReq (in TriComponentIdType component)	`void tciKillTestComponentReq` `   (TriComponentId component)`	

TBoolean tciTestComponentAliveReq
 (in TriComponentIdType component)
	`TBoolean`
	`tciTestComponentAliveReq`
	`   (TriComponentId component)`	

TBoolean tciTestComponentKilledReq
 (in TriComponentIdType component)
	`TBoolean`
	`tciTestComponentKilledReq`
	`   (TriComponentId component)`	

TCI-TL provided		

void tliTcExecute
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TciTestCaseIdType tcId,
 in TriParameterListType triPars,
 in TriTimerDurationType dur)
	`void tliTcExecute`
	`   (String am, int ts, String src,`
	`   int line, TriComponentId c,`
	`   TciTestCaseIdType tcId,`
	`   TriParameterList triPars,`
	`   TriTimerDuration dur)`	

void tliTcStart
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TciTestCaseIdType tcId,
 in TciParameterListType tciPars,
 in TriTimerDurationType dur)
	`void tliTcStart`
	`   (String am, int ts, String src,`
	`   int line, TriComponentId c,`
	`   TciTestCaseIdType tcId,`
	`   TriParameterListType tciPars,`
	`   TriTimerDuration dur)`	

void tliTcStop(in TString am, in TInteger ts,
 in TString src, in TInteger line,
 in TriComponentIdType c)
	`void tliTcStop`
	`   (String am, int ts, String src,`
	`   int line, TriComponentId c)`	

void tliTcStarted
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TciTestCaseIdType tcId,
 in TciParameterListType tciPars,
 in TriTimerDurationType dur)
	`void tliTcStarted`
	`   (String am, int ts, String src,`
	`   int line, TriComponentId c,`
	`   TciTestCaseIdType tcId,`
	`   TciParameterListType tciPars,`
	`   TriTimerDuration dur)`	

void tliTcTerminated
 (in TString am, in TInteger ts, in TString src,
 in TInteger line,
 in TriComponentIdType c,
 in TciTestCaseIdType tcId,
 in TciParameterListType tciPars,
 in VerdictValue verdict)
	`void tliTcTerminated`
	`   (String am, int ts, String src,`
	`   int line, TriComponentId c,`
	`   TciTestCaseIdType tcId,`
	`   TciParameterListType tciPars,`
	`   VerdictValue verdict)`	

void tliCtrlStart
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line,
 	
in TriComponentIdType c)
	```void tliCtrlStart```
	```(String am, int ts, String src,```
	```int line, TriComponentId c)```	

void tliCtrlStop
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c)
	```void tliCtrlStop```
	```(String am, int ts, String src,```
	```int line, TriComponentId c)```	

void tliCtrlTerminated
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c)
	```void tliCtrlTerminated```
	```(String am, int ts, String src,```
	```int line, TriComponentId c)```	

void tliMSend_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to, in Value msgValue,
 	
in TriAddressType address,
 	
in TciStatusType encoderFailure,
 	
in TriMessageType msg,
 	
in TriStatusType transmissionFailure)
	```void tliMSend_m```
	```(String am, int ts, String src,```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortId to,```
	```Value msgValue,```
	```TriAddress address,```
	```TciStatus encoderFailure,```
	```TriMessage msg,```
	```TriStatus transmissionFailure)```	

void tliMSend_m_BC
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to, in Value msgValue,
 	
in TciStatusType encoderFailure,
 	
in TriMessageType msg,
 	
in TriStatusType transmissionFailure)
	```void tliMSend_m_BC```
	```(String am, int ts, String src,```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortId to,```
	```Value msgValue,```
	```TciStatus encoderFailure,```
	```TriMessage msg,```
	```TriStatus transmissionFailure)```	

| void tliMSend_m_MC<br><br>(in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType to, in Value msgValue,<br><br> in TriAddressListType addresses,<br><br> in TciStatusType encoderFailure,<br><br> in TriMessageType msg,<br><br> in TriStatusType transmissionFailure) | ```
void tliMSend_m_MC
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId to,
 Value msgValue,
 TriAddressList addresses,
 TciStatus encoderFailure,
 TriMessage msg,
 TriStatus transmissionFailure)
``` | |
| void tliMSend_c<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType to, in Value msgValue,<br><br> in TriStatusType transmissionFailure) | ```
void tliMSend_c
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId to,
 Value msgValue,
 TriStatus transmissionFailure)
``` | |
| void tliMSend_c_BC<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdListType to,<br><br> in Value msgValue,<br><br> in TriStatusType transmissionFailure) | ```
void tliMSend_c_BC
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortIdList to,
 Value msgValue,
 TriStatus transmissionFailure)
``` | |
| void tliMSend_c_MC<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdListType to,<br><br> in Value msgValue,<br><br> in TriStatusType transmissionFailure) | ```
void tliMSend_c_MC
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortIdList to,
 Value msgValue,
 TriStatus transmissionFailure)
``` | |
| void tliMDetected_m<br><br>(in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType from, in TriMessageType msg,<br><br> in TriAddressType address) | ```
void tliMDetected_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId from,
 TriMessage msg,
 TriAddress address)
``` | |

void tliMDetected_c
 	
(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType from, in Value msgValue)
	```void tliMDetected_c```
	 ```(String am, int ts,```
	 ```String src, int line,```
	 ```TriComponentId c,```
	 ```TriPortId at, TriPortId from,```
	 ```Value msgValue)```	

void tliMMismatch_m
 	
(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in Value msgValue,
 	
 in TciValueTemplate msgTmpl,
 	
 in TciValueDifferenceList diffs,
 	
 in TriAddressType address,
 	
 in TciValueTemplate addressTmpl)
	```void tliMMismatch_m```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriPortId at,```
	 ```Value msgValue,```
	 ```TciValueTemplate msgTmpl,```
	 ```TciValueDifferenceList diffs,```
	 ```TriAddress address,```
	 ```TciValueTemplate addressTmpl)```	

void tliMMismatch_c
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in Value msgValue,
 	
 in TciValueTemplate msgTmpl,
 	
 in TciValueDifferenceList diffs,
 	
 in TriComponentIdType from,
 	
 in TciNonValueTemplate fromTmpl)
	```void tliMMismatch_c```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriPortId at,```
	 ```Value msgValue,```
	 ```TciValueTemplate msgTmpl,```
	 ```TciValueDifferenceList diffs,```
	 ```TriComponentId from,```
	 ```TciNonValueTemplate fromTmpl)```	

void tliMReceive_m
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in Value msgValue,
 	
 in TciValueTemplate msgTmpl,
 	
 in TriAddressType address,
 	
 in TciValueTemplate addressTmpl)
	```void tliMReceive_m(String am,```
	 ```int ts, String src, int line,```
	 ```TriComponentId c,```
	 ```TriPortId at,```
	 ```Value msgValue,```
	 ```TciValueTemplate msgTmpl,```
	 ```TriAddress address,```
	 ```TciValueTemplate addressTmpl)```	

void tliMReceive_c
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in Value msgValue,
 	
 in TciValueTemplate msgTmpl,
 	
 in TriComponentIdType fromComp,
 	
 in TciNonValueTemplate fromTmpl)
	```void tliMReceive_c```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriPortId at,```
	 ```Value msgValue,```
	 ```TciValueTemplate msgTmpl,```
	 ```TriComponentId fromComp,```
	 ```TciNonValueTemplate fromTmpl)```	

void tliPrCall_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in TriAddressType address, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriStatusType transmissionFailure)	```void tliPrCall_m (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature, TciParameterListType tciPars, TriAddress address, TriStatus encoderFailure, TriParameterList triPars, TciStatus transmissionFailure)```	
void tliPrCall_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriStatusType transmissionFailure)	```void tliPrCall_m_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to, TriSignatureId signature, TciParameterListType tciPars, TciStatus encoderFailure, TriParameterList triPars, TriStatus transmissionFailure)```	
void tliPrCall_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in TriAddressListType addresses, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriStatusType transmissionFailure)	```void tliPrCall_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to, TriSignatureId signature, TciParameterListType tciPars, TriAddressList addresses, TciStatus encoderFailure, TriParameterList triPars, TriStatus transmissionFailure)```	
void tliPrCall_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in TriStatusType transmissionFailure)	```void tliPrCall_c (String am, int ts, String src int line, TriComponentId c, TriPortId at, TriPortId to TriSignatureId signature, TciParameterListType tciPars, TriStatus transmissionFailure)```	

void tliPrCall_c_BC
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdListType to,
 	
 in TriSignatureIdType signature,
 	
 in TciParameterListType tciPars,
 	
 in TriStatusType transmissionFailure)
	```void tliPrCall_c_BC```
	```(String am, int ts, String src```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortIdList to,```
	```TriSignatureId signature,```
	```TciParameterListType tciPars,```
	```TriStatus transmissionFailure)```	

void tliPrCall_c_MC
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdListType to,
 	
 in TriSignatureIdType signature,
 	
 in TciParameterListType tciPars,
 	
 in TriStatusType transmissionFailure)
	```void tliPrCall_c_MC```
	```(String am, int ts, String src```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortIdList to,```
	```TriSignatureId signature,```
	```TciParameterListType tciPars,```
	```TriStatus transmissionFailure)```	

void tliPrGetCallDetected_m
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType from, in TriSignatureIdType signature,
 	
 in TriParameterListType triPars,
 	
 in TriAddressType address)
	```void tliPrGetCallDetected_m```
	```(String am, int ts, String src,```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortId from,```
	```TriSignatureId signature,```
	```TriParameterList triPars,```
	```TriAddress address)```	

void tliPrGetCallDetected_c
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType from,
 	
 in TriSignatureIdType signature,
 	
 in TciParameterListType tciPars)
	```void tliPrGetCallDetected_c```
	```(String am, int ts, String src,```
	```int line, TriComponentId c,```
	```TriPortId at, TriPortId from,```
	```TriSignatureId signature,```
	```TciParameterListType tciPars)```	

| void tliPrGetCallMismatch_m<br>(in TString am, in TInteger ts, in TString src,<br> in TInteger line, in TriComponentIdType c,<br> in TriPortIdType at,<br> in TriSignatureIdType signature,<br> in TciParameterListType tciPars,<br> in TciValueTemplate parsTmpl,<br> in TciValueDifferenceList diffs,<br> in TriAddressType address,<br> in TciValueTemplate addressTmpl) | ```
void tliPrGetCallMismatch_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature,
 TciParameterListType tciPars,
 TciValueTemplate parsTmpl,
 TciValueDifferenceList diffs,
 TriAddress address,
 TciValueTemplate addressTmpl)
``` | |
| void tliPrGetCallMismatch_c<br> (in TString am, in TInteger ts, in TString src,<br> in TInteger line, in TriComponentIdType c,<br> in TriPortIdType at,<br> in TriSignatureIdType signature,<br> in TciParameterListType tciPars,<br> in TciValueTemplate parsTmpl,<br> in TciValueDifferenceList diffs,<br> in TriComponentIdType from,<br> in TciNonValueTemplate fromTmpl) | ```
void tliPrGetCallMismatch_c
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature,
 TciParameterListType tciPars,
 TciValueTemplate parsTmpl,
 TciValueDifferenceList diffs,
 TriComponentId from,
 TciNonValueTemplate fromTmpl)
``` | |
| void tliPrGetCall_m<br>(in TString am, in TInteger ts, in TString src,<br> in TInteger line, in TriComponentIdType c,<br> in TriPortIdType at,<br> in TriSignatureIdType signature,<br> in TciParameterListType tciPars,<br> in TciValueTemplate parsTmpl,<br> in TriAddressType address,<br> in TciValueTemplate addressTmpl) | ```
void tliPrGetCall_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature,
 TciParameterListType tciPars,
 TciValueTemplate parsTmpl,
 TriAddress address,
 TciValueTemplate addressTmpl)
``` | |

void tliPrGetCall_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	```void tliPrGetCall_c```
	```(String am, int ts, String src,```
	``` int line, TriComponentId c,```
	``` TriPortId at,```
	``` TriSignatureId signature,```
	``` TciParameterListType tciPars,```
	``` TciValueTemplate parsTmpl,```
	``` TriComponentId from,```
	``` TciNonValueTemplate fromTmpl)```	

void tliPrReply_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value replValue,
 	
in TriAddressType address,
 	
in TciStatusType encoderFailure,
 	
in TriParameterListType triPars,
 	
in TriParameterType repl,
 	
in TriStatusType transmissionFailure)
	```void tliPrReply_m```
	```(String am, int ts, String src,```
	``` int line, TriComponentId c,```
	``` TriPortId at, TriPortId to,```
	``` TriSignatureIdType signature,```
	``` TciParameterListType tciPars,```
	``` Value replValue,```
	``` TriAddress address,```
	``` TciStatus encoderFailure,```
	``` TriParameterList triPars,```
	``` TriParameter repl,```
	``` TriStatus transmissionFailure)```	

void tliPrReply_m_BC
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value replValue,
 	
in TciStatusType encoderFailure,
 	
In TriParameterListType triPars,
 	
in TriParameterType repl,
 	
in TriStatusType transmissionFailure)
	```void tliPrReply_m_BC```
	```(String am, int ts, String src,```
	``` int line, TriComponentId c,```
	``` TriPortId at, TriPortId to,```
	``` TriSignatureIdType signature,```
	``` TciParameterListType tciPars,```
	``` Value replValue,```
	``` TciStatus encoderFailure,```
	``` TriParameterList triPars,```
	``` TriParameter repl,```
	``` TriStatus transmissionFailure)```	

void tliPrReply_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value replValue, in TriAddressListType addresses, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriParameterType repl, in TriStatusType transmissionFailure)	``` void tliPrReply_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to, TriSignatureIdType signature, TciParameterListType tciPars, Value replValue, TriAddressList addresses, TriStatus encoderFailure, TriParameterList triPars, TriParameter repl, TciStatus transmissionFailure) ```	
void tliPrReply_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value replValue, in TriStatusType transmissionFailure)	``` void tliPrReply_c (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to, TriSignatureId signature, TciParameterListType tciPars, Value replValue, TriStatus transmissionFailure) ```	
void tliPrReply_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value replValue, in TriStatusType transmissionFailure)	``` void tliPrReply_c_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterListType tciPars, Value replValue, TriStatus transmissionFailure) ```	

| void tliPrReply_c_MC<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdListType to,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in Value replValue,<br><br> in TriStatusType transmissionFailure) | ```
void tliPrReply_c_MC
(String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortIdList to,
 TriSignatureId signature,
 TciParameterListType tciPars,
 Value replValue,
 TriStatus transmissionFailure)
``` | |
| void tliPrGetReplyDetected_m<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType from,<br><br> in TriSignatureIdType signature,<br><br> in TriParameterListType triPars,<br><br> in TriParameterType repl,<br><br> in TriAddressType address) | ```
void tliPrGetReplyDetected_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId from,
 TriSignatureId signature,
 in TriParameterList triPars,
 TriParameter repl,
 TriAddress address)
``` | |
| void tliPrGetReplyDetected_c<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType from,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in Value replValue) | ```
void tliPrGetReplyDetected_c
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, in TriPortId
from,
 TriSignatureId signature,
 in TciParameterListType
tciPars,
 Value replValue)
``` | |
| void tliPrGetReplyMismatch_m<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in TciValueTemplate parsTmpl,<br><br> in Value replValue,<br><br> in TciValueTemplate replyTmpl,<br><br> in TciValueDifferenceList diffs,<br><br> in TriAddressType address,<br><br> in TciValueTemplate addressTmpl) | ```
void tliPrGetReplyMismatch_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature,
 TciParameterListType tciPars,
 in TciValueTemplate parsTmpl,
 Value replValue,
 TciValueTemplate replyTmpl,
 TciValueDifferenceList diffs,
 TriAddress address,
 TciValueTemplate addressTmpl)
``` | |

void tliPrGetReplyMismatch_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in Value replValue,
 	
in TciValueTemplate replyTmpl,
 	
in TciValueDifferenceList diffs,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	`void tliPrGetReplyMismatch_c`
	`(String am, int ts, String src,`
	`int line, TriComponentId c,`
	`TriPortId at,`
	`TriSignatureId signature,`
	`TciParameterListType tciPars,`
	`in TciValueTemplate parsTmpl,`
	`Value replValue,`
	`TciValueTemplate replyTmpl,`
	`TciValueDifferenceList diffs,`
	`TriComponentId from,`
	`TciNonValueTemplate fromTmpl)`	

void tliPrGetReply_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in Value replValue,
 	
in TciValueTemplate replyTmpl,
 	
in TriAddressType address,
 	
in TciValueTemplate addressTmpl)
	`void tliPrGetReply_m`
	`(String am, int ts, String src,`
	`int line, TriComponentId c,`
	`TriPortId at,`
	`TriSignatureId signature,`
	`TciParameterListType tciPars,`
	`in TciValueTemplate parsTmpl,`
	`Value replValue,`
	`TciValueTemplate replyTmpl,`
	`TriAddress address,`
	`TciValueTemplate addressTmpl)`	

void tliPrGetReply_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in Value replValue,
 	
in TciValueTemplate replyTmpl,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	`void tliPrGetReply_c`
	`(String am, int ts, String src,`
	`int line, TriComponentId c,`
	`TriPortId at,`
	`TriSignatureId signature,`
	`TciParameterListType tciPars,`
	`in TciValueTemplate parsTmpl,`
	`Value replValue,`
	`TciValueTemplate replyTmpl,`
	`TriComponentId from,`
	`TciNonValueTemplate fromTmpl)`	

| void tliPrRaise_m<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType to,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in Value excValue, in TriAddressType address,<br><br> in TciStatusType encoderFailure,<br><br> in TriExceptionType exc,<br><br> in TriStatusType transmissionFailure) | ```
void tliPrRaise_m
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId to,
 TriSignatureId signature,
 TciParameterListType tciPars,
 Value excValue,
 TriAddress address,
 TciStatus encoderFailure,
 TriException exc,
 TriStatus transmissionFailure)
``` | |
| void tliPrRaise_m_BC<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType to,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in Value excValue,<br><br> in TciStatusType encoderFailure,<br><br> in TriExceptionType exc,<br><br> in TriStatusType transmissionFailure) | ```
void tliPrRaise_m_BC
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId to,
 TriSignatureId signature,
 TciParameterListType tciPars,
 Value excValue,
 TciStatus encoderFailure,
 TriException exc,
 TriStatus transmissionFailure)
``` | |
| void tliPrRaise_m_MC<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at, in TriPortIdType to,<br><br> in TriSignatureIdType signature,<br><br> in TciParameterListType tciPars,<br><br> in Value excValue,<br><br> in TriAddressListType addresses,<br><br> in TciStatusType encoderFailure,<br><br> in TriExceptionType exc,<br><br> in TriStatusType transmissionFailure) | ```
void tliPrRaise_m_MC
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at, TriPortId to,
 TriSignatureId signature,
 TciParameterListType tciPars,
 Value excValue,
 TriAddressList addresses,
 TciStatus encoderFailure,
 TriException exc,
 TriStatus transmissionFailure)
``` | |

void tliPrRaise_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value excValue, in TriStatusType transmissionFailure)	```void tliPrRaise_c`` ``(String am, int ts, String src,`` ``int line, TriComponentId c,`` ``TriPortId at, TriPortId to,`` ``TriSignatureId signature,`` ``TciParameterListType tciPars,`` ``Value excValue,`` ``TriStatus transmissionFailure)```	
void tliPrRaise_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value excValue, in TriStatusType transmissionFailure)	```void tliPrRaise_c_BC`` ``(String am, int ts, String src,`` ``int line, TriComponentId c,`` ``TriPortId at, TriPortIdList to,`` ``TriSignatureId signature,`` ``TciParameterListType tciPars,`` ``Value excValue,`` ``TriStatus transmissionFailure)```	
void tliPrRaise_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value excValue, in TriStatusType transmissionFailure)	```void tliPrRaise_c_MC`` ``(String am, int ts, String src,`` ``int line, TriComponentId c,`` ``TriPortId at, TriPortIdList to,`` ``TriSignatureId signature,`` ``TciParameterListType tciPars,`` ``Value excValue,`` ``TriStatus transmissionFailure)```	
void tliPrCatchDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)	```void tliPrCatchDetected_m`` ``(String am, int ts, String src,`` ``int line, TriComponentId c,`` ``TriPortId at, TriPortId from,`` ``TriSignatureId signature,`` ``TriException exc,`` ``TriAddress address)```	

void tliPrCatchDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from, in TriSignatureIdType signature, in Value excValue)	```void tliPrCatchDetected_c (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from, TriSignatureId signature, Value excValue)```	
void tliPrCatchMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	```void tliPrCatchMismatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature, Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, TriAddress address, TciValueTemplate addressTmpl)```	
void tliPrCatchMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	```void tliPrCatchMismatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature, Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, TriComponentId from, TciNonValueTemplate fromTmpl)```	
void tliPrCatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	```void tliPrCatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature, Value excValue, TciValueTemplate excTmpl, TriAddress address, TciValueTemplate addressTmpl)```	

| void tliPrCatch_c<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at,<br><br> in TriSignatureIdType signature,<br><br> in Value excValue,<br><br> in TciValueTemplate excTmpl,<br><br> in TriComponentIdType from,<br><br> in TciNonValueTemplate fromTmpl) | ```
void tliPrCatch_c
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature,
 Value excValue,
 TciValueTemplate excTmpl,
 TriComponentId from,
 TciNonValueTemplate fromTmpl)
``` | |
| void tliPrCatchTimeoutDetected<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at,<br><br> in TriSignatureIdType signature) | ```
void tliPrCatchTimeoutDetected
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature)
``` | |
| void tliPrCatchTimeout<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriPortIdType at,<br><br> in TriSignatureIdType signature) | ```
void tliPrCatchTimeout
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriPortId at,
 TriSignatureId signature)
``` | |
| void tliCCreate<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriComponentIdType comp,<br><br> in TString name, TBoolean alive) | ```
void tliCCreate
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriComponentId comp,
 String name, Boolean alive)
``` | |
| void tliCStart<br><br> (in TString am, in TInteger ts, in TString src,<br><br> in TInteger line, in TriComponentIdType c,<br><br> in TriComponentIdType comp,<br><br> in TciBehaviourIdType name,<br><br> in TciParameterListType tciPars) | ```
void tliCStart
 (String am, int ts, String src,
 int line, TriComponentId c,
 TriComponentId comp,
 TciBehaviourIdType name,
 TciParameterListType tciPars)
``` | |

void tliCRunning
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriComponentIdType comp,
 	
 in ComponentStatusType status)
	```void tliCRunning```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriComponentId comp,```
	 ```ComponentStatus status)```	

void tliCStop
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriComponentIdType comp)
	```void tliCStop```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriComponentId comp)```	

void tliCDoneMismatch
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriComponentIdType comp,
 	
 in TciNonValueTemplate compTmpl)
	```void tliCDoneMismatch```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriComponentId comp,```
	 ```TciNonValueTemplate compTmpl)```	

void tliCDone
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TciNonValueTemplate compTmpl)
	```void tliCDone```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TciNonValueTemplate compTmpl)```	

void tliCTerminated
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in VerdictValue verdict)
	```void tliCTerminated```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```VerdictValue verdict)```	

void tliPConnect
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType port1,
 	
 in TriPortIdType port2)
	```void tliPConnect```
	 ```(String am, int ts, String src,```
	 ```int line, TriComponentId c,```
	 ```TriPortId port1,```
	 ```TriPortId port2)```	

void tliPDisconnect
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriComponentIdType c1,
 	
 in TriPortIdType port1,
 	
 in TriComponentIdType c2,
 	
 in TriPortIdType port2)
	`void tliPDisconnect`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriComponentId c1,`
	`  TriPortId port1,`
	`  TriComponentId c2,`
	`  TriPortId port2)`	

void tliPMap
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType port1,
 	
 in TriPortIdType port2)
	`void tliPMap`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriPortId port1,`
	`  TriPortId port2)`	

void tliPUnmap
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriComponentIdType c1,
 	
 in TriPortIdType port1,
 	
 in TriComponentIdType c2,
 	
 in TriPortIdType port2)
	`void tliPUnmap`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriComponentId c1,`
	`  TriPortId port1,`
	`  TriComponentId c2,`
	`  TriPortId port2)`	

void tliPClear
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType port)
	`void tliPClear`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriPortId port)`	

void tliPStart(in TString am, in TInteger ts,
 	
 in TString src, in TInteger line,
 	
 in TriComponentIdType c,
 	
 in TriPortIdType port)
	`void tliPStart`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriPortId port)`	

void tliPStop
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType port)
	`void tliPStop`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriPortId port)`	

void tliPHalt
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType port)
	`void tliPHalt`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriPortId port)`	

void tliEncode
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in Value val, in TciStatusType encoderFailure,
 	
 in TriMessageType msg, in TString codec)
	`void tliEncode`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  Value val,`
	`  TciStatus encoderFailure,`
	`  TriMessage msg, String codec)`	

void tliDecode
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriMessageType msg,
 	
 in TciStatusType decoderFailure, in Value val,
 	
 in TString codec)
	`void tliDecode`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriMessage msg,`
	`  TciStatus decoderFailure,`
	`  Value val, String codec)`	

void tliTTimeoutDetected
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriTimerIdType timer)
	`void tliTTimeoutDetected`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer)`	

void tliTTimeoutMismatch
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriTimerIdType timer,
 	
 in TciNonValueTemplate timerTmpl)
	`void tliTTimeoutMismatch`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer,`
	`  TciNonValueTemplate timerTmpl)`	

void tliTTimeout
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriTimerIdType timer,
 	
 in TciNonValueTemplate timerTmpl)
	`void tliTTimeout`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerIdType timer,`
	`  TciNonValueTemplate timerTmpl)`	

void tliTStart
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriTimerIdType timer,
 in TriTimerDurationType dur)
	`void tliTStart`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer,`
	`  TriTimerDuration dur)`	

void tliTStop
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriTimerIdType timer,
 in TriTimerDurationType dur)
	`void tliTStop`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer,`
	`  TriTimerDuration dur)`	

void tliTRead
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriTimerIdType timer,
 in TriTimerDurationType elapsed)
	`void tliTRead`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer,`
	`  TriTimerDuration elapsed)`	

void tliTRunning
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriTimerIdType timer, in TimerStatusType status)
	`void tliTRunning`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  TriTimerId timer,`
	`  TimerStatus status)`	

void tliSEnter
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in QualifiedName name,
 in TciParameterListType tciPars,
 in TString kind)
	`void tliSEnter`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  QualifiedName name,`
	`  TciParameterList tciPars,`
	`  String kind)`	

void tliSLeave
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in QualifiedName name,
 in TciParameterListType tciPars,
 in Value returnValue, in TString kind)
	`void tliSLeave`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  QualifiedName name,`
	`  TciParameterListType tciPars,`
	`  Value returnValue,`
	`  String kind)`	

void tliVar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value varValue)	```void tliVar (String am, int ts, String src, int line, TriComponentId c, QualifiedName name, Value varValue)```	
void tliModulePar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value parValue)	```void tliModulePar (String am, int ts, String src, int line, TriComponentId c, QualifiedName name, Value parValue)```	
void tliGetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	```void tliGetVerdict (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)```	
void tliSetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	```void tliSetVerdict (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)```	
void tliLog (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString log)	```void tliLog (String am, int ts, String src, int line, TriComponentId c, String log)```	
void tliAEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	```void tliAEnter (String am, int ts, String src, int line, TriComponentId c)```	
void tliALeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	```void tliALeave (String am, int ts, String src, int line, TriComponentId c)```	
void tliADefaults (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	```void tliADefaults (String am, int ts, String src, int line, TriComponentId c)```	

void tliAActivate
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in QualifiedName name,
 	
 in TciParameterListType tciPars, in Value ref)
	`void tliAActivate`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  QualifiedName name,`
	`  TciParameterListType tciPars,`
	`  Value ref)`	

void tliADeactivate
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in Value ref)
	`void tliADeactivate`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  Value ref)`	

void tliANomatch
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c)
	`void tliANomatch`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c)`	

void tliARepeat
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c)
	`void tliARepeat`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c)`	

void tliAWait
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c)
	`void tliAWait`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c)`	

void tliAction
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TString action)
	`void tliAction`
	`  (String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  String action)`	

void tliMatch
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in Value expr, in TciValueTemplate tmpl)
	`void tliMatch`
	`(String am, int ts, String src,`
	`  int line, TriComponentId c,`
	`  Value expr, TciValueTemplate`
	`tmpl)`	

void tliMatchMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value expr, in TciValueTemplate tmpl, in TciValueDifferenceList diffs)	`void tliMatchMismatch (String am, int ts, String src, int line, TriComponentId c, Value expr, TciValueTemplate tmpl, TciValueDifferenceList diffs)`	
void tliInfo (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TInteger level, in TString info)	`void tliInfo (in String am, int ts, in String src, int line, in TriComponentId c, in int level, in String info)`	

9.5 Data

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
TciModuleIdType	**QualifiedName**	

TciModuleParameterType	`typedef struct TciModuleParameterType`
	`{`
	`   String   parName;`
	`   TciValue defaultValue;`
	`} TciModuleParameterType;`	

TciModuleParameterListType	`typedef struct TciModuleParameterListType`
	`{`
	`   long int              length;`
	`   TciModuleParameterType *modParList;`
	`} TciModuleParameterListType;`	

TciParameterType	`typedef struct TciParameterType`
	`{`
	`   String   parName;`
	`   TciParameterPassingModeType parPassMode;`
	`   TciValue parValue;`
	`} TciParameterType;`	

TciParameterPassingModeType	`typedef enum`
	`{`
	`   TCI_IN_PAR    = 0,`
	`   TCI_INOUT_PAR = 1,`
	`   TCI_OUT_PAR   = 2`
	`} TciParameterPassingModeType;`	

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
| TciParameterListType | ```
typedef struct
TciParameterListType
{
  long int        length;
  TciParameterType *parList;
} TciParameterListType;
``` | length 0 shall be interpreted as "empty list". |
| TciParameterTypeListType | ```
typedef struct
TciParameterTypeListType
{
  long int  length;
  TciType  *parList;
} TciParameterTypeListType;
``` | length 0 shall be interpreted as "empty list". |
| TciTestCaseIdListType | ```
typedef struct
TciTestCaseIdListType
{
  long int        length;
  QualifiedName *idList;
} TciTestCaseIdListType;
``` | length 0 shall be interpreted as "empty list". |
| TciTypeClassType | ```
typedef enum
{
  TCI_ADDRESS_TYPE,
  TCI_ANYTYPE_TYPE,
  TCI_BITSTRING_TYPE,
  TCI_BOOLEAN_TYPE,
  TCI_CHAR_TYPE,
  TCI_CHARSTRING_TYPE,
  TCI_COMPONENT_TYPE,
  TCI_ENUMERATED_TYPE,
  TCI_FLOAT_TYPE,
  TCI_HEXSTRING_TYPE,
  TCI_INTEGER_TYPE,
  TCI_OBJID_TYPE,
  TCI_OCTETSTRING_TYPE,
  TCI_RECORD_TYPE,
  TCI_RECORD_OF_TYPE,
  TCI_SET_TYPE,
  TCI_SET_OF_TYPE,
  TCI_UNION_TYPE,
  TCI_UNIVERSAL_CHAR_TYPE,
  TCI_UNIVERSAL_CHARSTRING_TYPE,
  TCI_VERDICT_TYPE
} TciTypeClassType;
``` | |

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
| TciTestComponentKindType | ```
typedef enum
{
    TCI_CTRL_COMP,
    TCI_MTC_COMP,
    TCI_PTC_COMP,
    TCI_SYS_COMP,
    TCI_ALIVE_COMP
} TciTestComponentKindType;
``` | |
| **TciBehaviourIdType** | **QualifiedName** | |
| TciValueDifference | ```
typedef struct TciValueDifference
{
    TciValue val;
    TciValueTemplate tmpl;
    String desc;
} TciValueDifference;
``` | |
| TciValueDifferenceList | ```
typedef struct
TciValueDifferenceList
{
    long int        length;
    TciValueDifference[] diffList;
} TciValueDifferenceList;
``` | length 0 shall be interpreted as "empty list". |

9.6   Miscellaneous

TCI concept	ANSI C representation	Notes and comments
Verdict representation		
NONE	`const int TCI_VERDICT_NONE   = 0`	Since the TciVerdictValue interface is defined in terms of integers, consensus must be established on which value defines which verdict.
PASS	`const int TCI_VERDICT_PASS   = 1`	
INCONC	`const int TCI_VERDICT_INCONC = 2`	
FAIL	`const int TCI_VERDICT_FAIL   = 3`	
ERROR	`const int TCI_VERDICT_ERROR  = 4`	
Objid representation		
| Objid | ```
typedef struct TciObjidValue
{
    long int     length;
    TciObjidElem *elements;
} TciObjidValue;
``` | Since the Objid value is returned "as is" via the Objid value interface, a representation must be defined. |

TCI concept	ANSI C representation	Notes and comments
TciObjidElem	```typedef struct TciObjidElemValue
{
char*   elem_as_ascii;
long int elem_as_number;
void*   aux;
} TciObjidElemValue;```	
CharstringValue representation		
TciCharString	```typedef struct TciCharStringValue
{
unsigned long int length;
char*   string;
} TciCharStringValue```	
Universal Character[string] representation		
Universal Char	```typedef unsigned char
TciUCValue```	
Universal Charstring	```typedef struct TciUCStringValue
{
unsigned long int length;
TciUCType   *string;
} TciUCStringValue;```	

10   W3C XML mapping

10.1   Introduction

This clause introduces the TCI XML mapping [W3C Primer], [W3C structures] and [W3C Datatypes] for the logging interface of TCI. The XML mapping for the logging interface defines how the IDL definitions described in clause 7 are mapped to XML. The schema definitions for this mapping are given in Annex B.

10.2   Scopes

The IDL module `tciInterface` is mapped to an XML schema with the name space `http://uri.etsi.org/TTCN-3/tci/TLI_v3_2_1.xsd`

This schema uses further schemas:

- `http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd`

  for the mapping of simple types to XML.

- `http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd`

  for the mapping of structured types to XML.

- `http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd`

  for the mapping of values to XML.

- `http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd`

  for the mapping of templates to XML.

- `http://uri.etsi.org/TTCN-3/tci/Events_v3_2_1.xsd`

  for the mapping of logging events to XML.

10.3 Type mapping

10.3.1 Mapping of simple types

10.3.1.1 TBoolean

The IDL **TBoolean** type is mapped to the xsd basic type `boolean`.

10.3.1.2 TString

The IDL **TString** type is mapped to the xsd basic type `string`.

10.3.1.3 TInteger

The IDL **TInteger** type is mapped to the xsd basic type `integer`.

10.3.1.4 TriTimerDurationType

The IDL **TriTimerDurationType** type is mapped to the xsd basic type `float`.

10.3.1.5 TciParameterPassingModeType

The IDL **TciParameterPassingModeType** type is mapped to the xsd basic type `string` with enumeration values `'in'`, `'out'` and `'inout'`.

10.3.1.6 TriStatusType

The IDL **TriStatusType** type is mapped to the xsd basic type `string` with enumeration values `'TRI_Ok'` and `'TRI_Error'`.

10.3.1.7 TciStatusType

The IDL **TriStatusType** type is mapped to the xsd basic type `string` with enumeration values `'TCI_Ok'` and `'TCI_Error'`.

10.3.1.8 ComponentStatusType

The IDL **ComponentStatusType** type is mapped to the xsd basic type `string` with enumeration values `'inactiveC'`, `'runningC'`, `'stoppedC'` and `'killedC'`.

10.3.1.9 TimerStatusType

The IDL **TimerStatusType** type is mapped to the xsd basic type `string` with enumeration values `'runningT'`, `'inactiveT'` and `'expiredT'`.

10.3.1.10 PortStatusType

The IDL **PortStatusType** type is mapped to the xsd basic type `string` with enumeration values `'startedP'`, `'haltedP'` and `'stoppedP'`.

10.3.2 Complex type mapping

10.3.2.1 TriPortIdType

TriPortIdType is mapped to the following complex type:

```
<xsd:complexType name="TriPortIdType">
    <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType" />
        <xsd:element name="port" type="Types:Port"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- `comp`   The TRI component identifier.
- `port`   The identification of the port.

Attributes:

- none.

10.3.2.2   TriComponentIdType

TriComponentIdType is mapped to the following complex type:

```
<xsd:compleporxType name="TriComponentIdType">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="null" type="Templates:null"/>
            <xsd:element name="id" type="Types:Id"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- id   The identifier of the TRI component.

- null   The null identifier. To be used if there is no TRI component identifier.

Attributes:

- none.

10.3.2.3   TriComponentIdListType

TriComponentIdListType is mapped to the following complex type:

```
<xsd:complexType name="TriComponentIdListType">
    <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- comp   The identifiers of TRI components in that list.

Attributes:

- none.

10.3.2.4   Port

Port is mapped to the following complex type:

```
<xsd:complexType name="Port">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id"/>
        <xsd:element name="index" type="xsd:int"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- id   The port identifier.

- port   The port index.

Attributes:

- none.

10.3.2.5 Id

Id is used as identification for components, ports and timers and is mapped to the following complex type:

```xsd
<xsd:complexType name="Id">
    <xsd:sequence>
        <xsd:element name="name" type="SimpleTypes:TString"/>
        <xsd:element name="id" type="SimpleTypes:TString"/>
        <xsd:element name="type" type="SimpleTypes:TString"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- name   The name of the component, port or timer.

- id   The internal representation of the component, port or timer.

- type   The type of the component, port or timer.

Attributes:

- none.

10.3.2.6 TriMessageType

TriMessageType is mapped to the following complex type:

```xsd
<xsd:complexType name="TriMessageType">
    <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>
```

Elements:

- val   The encoded message.

Attributes:

- none.

10.3.2.7 TriParameterType

TriParameterType is mapped to the following complex type:

```xsd
<xsd:complexType name="TriParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="xsd:hexBinary"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="SimpleTypes:TString"/>
    <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

Elements:

- val   The encoded parameter.

Attributes:

- name   The parameter name.

- mode   The parameter passing mode.

10.3.2.8   TriParameterListType

TriParameterListType is mapped to the following complex type:

```
<xsd:complexType name="TriParameterListType">
    <xsd:sequence>
        <xsd:element name="par" type="Types:TriParameterType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

Sequence of Elements:

- par   The parameters in that list.

Attributes:

- none.

10.3.2.9   TriAddressType

TriAddressType is mapped to the following complex type:

```
<xsd:complexType name="TriAddressType">
    <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>
```

Elements:

- val   The address value.

Attributes:

- none.

10.3.2.10   TriAddressListType

TriAddressListType is mapped to the following complex type:

```
<xsd:complexType name="TriAddressListType">
    <xsd:sequence>
        <xsd:element name="addr" type="Types:TriAddressType" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

- addr   The addresses in that list.

Attributes:

- none.

10.3.2.11   TriExceptionType

TriExceptionType is mapped to the following complex type:

```
<xsd:complexType name="TriExceptionType">
    <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>
```

Elements:

- val   The exception.

Attributes:

- none.

10.3.2.12 TriSignatureIdType

TriSignatureIdType is mapped to the following complex type:

```
<xsd:complexType name="TriSignatureIdType">
    <xsd:attribute name="val" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

•   val   The signature.

Attributes:

•   none.

10.3.2.13 TriAddressType

TriAddressType is mapped to the following complex type:

```
<xsd:complexType name="TriAddressType">
    <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>
```

Elements:

•   val   The address within the SUT.

Attributes:

•   none.

10.3.2.14 TriTimerIdType

TriTimerIdType is mapped to the following complex type:

```
<xsd:complexType name="TriTimerIdType">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

•   id   The identification of the timer.

Attributes:

•   none.

10.3.2.15 TriTimerDurationType

TriTimerDurationType is mapped to the following simple type:

```
<xsd:simpleType name="TriTimerDurationType">
    <xsd:restriction base="xsd:float"/>
</xsd:simpleType>
```

10.3.2.16 QualifiedName

QualifiedName is used to fully qualify module parameters, variables, etc. and is mapped to the following complex type:

```
<xsd:complexType name="QualifiedName">
    <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
    <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

-       `moduleName` The module name of the TTCN-3 module.

-       `baseName`   The name of the object that is fully qualified.

Attributes:

-       none.

10.3.2.17 TciBehaviourIdType

`TciBehaviourIdType` is mapped to the following complex type:

```
<xsd:complexType name="TciBehaviourIdType">
    <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

-       `name`   The qualified name of the behaviour.

Attributes:

-       none.

10.3.2.18 TciTestCaseIdType

`TciTestCaseIdType` is mapped to the following complex type:

```
<xsd:complexType name="TciTestCaseIdType">
    <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName"/>
    </xsd:sequence>
</xsd:complexType>
```

Elements:

-       `name`   The qualified name of the test case.

Attributes:

-       none.

10.3.2.19 TciParameterType

`TciParameterType` is mapped to the following complex type:

```
<xsd:complexType name="TciParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="Values:Value"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="SimpleTypes:TString"/>
    <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

Elements:

-       `val`   The encoded parameter.

Attributes:

-       `name`   The parameter name.

-       `mode`   The parameter passing mode.

10.3.2.20 TciParameterListType

TciParameterListType is mapped to the following complex type:

```
<xsd:complexType name="TciParameterListType">
    <xsd:sequence>
        <xsd:element name="par" type="Types:TciParameterType"
         minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

Sequence of elements:

- `par`   The parameters in that list.

Attributes:

- none.

10.3.3 Abstract value mapping

10.3.3.1 Value

Value is mapped to the following complex type:

```
<xsd:complexType name="Value" mixed="true">
    <xsd:choice>
        <xsd:element name="integer" type="Values:IntegerValue"/>
        <xsd:element name="float" type="Values:FloatValue"/>
        <xsd:element name="boolean" type="Values:BooleanValue"/>
        <xsd:element name="objid" type="Values:ObjidValue"/>
        <xsd:element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:CharstringValue"/>
        <xsd:element name="universal_charstring" type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
        <xsd:element name="anytype" type="Values:AnytypeValue"/>
        <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:attributeGroup name="ValueAtts">
    <xsd:attribute name="name" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="type" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="module" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="annotation" type="SimpleTypes:TString" use="optional"/>
</xsd:attributeGroup>
```

Choice of elements:

- `integer`   An integer value.

- `float`   A float value.

- `boolean`   A boolean value.

- `objid`   An objid value.

- `verdicttype`   A verdicttype value.

- `bitstring`   A bitstring value.

- `hexstring`   An hexstring value.

- `octetstring`   An octetstring value.

- `charstring`   A charstring value.

- •   `universal_charstring`   A universal charstring value.
- •   `record`   A record value.
- •   `record_of`   A record of value.
- •   `set`   A set value.
- •   `set_of`   A set of value.
- •   `enumerated`   An enumerated value.
- •   `union`   A union value.
- •   `anytype`   An anytype value.
- •   `address`   An address value.

Attributes:

- •   `name`   The name of the value, if known.
- •   `type`   The type of the value, if known.
- •   `module`   The module of the value, if known.
- •   `annotation` A helper attribute to provide additional matching/mismatching information, etc.

10.3.3.2   IntegerValue

`IntegerValue` is mapped to the following complex type:

```
<xsd:complexType name="IntegerValue">
   <xsd:choice >
      <xsd:element name="value" type="SimpleTypes:TString"/>
      <xsd:element name="null" type="Templates:null"/>
      <xsd:element name="omit" type="Templates:omit"/>
   </xsd:choice>
   <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Simple Content:

- •   `value`   The integer value as string.
- •   `null`   If no value is given.
- •   `omit`   If the value is omitted.

Attributes:

- •   The same attributes as those of Value.

10.3.3.3   FloatValue

`FloatValue` is mapped to the following complex type:

```
<xsd:complexType name="FloatValue">
   <xsd:simpleContent>
      <xsd:extension base="SimpleTypes:TString">
            <xsd:attributeGroup ref="Values:ValueAtts"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

-     `value`   The float value as string.

-     `null`   If no value is given.

-     `omit`   If the value is omitted.

Attributes:

-     The same attributes as those of Value.

10.3.3.4   BooleanValue

`BooleanValue` is mapped to the following complex type:

```
<xsd:complexType name="BooleanValue">
    <xsd:simpleContent>
       <xsd:extension base="SimpleTypes:TString">
             <xsd:attributeGroup ref="Values:ValueAtts"/>
       </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

-     `value`   The boolean value as string.

-     `null`   If no value is given.

-     `omit`   If the value is omitted.

Attributes:

-     The same attributes as those of Value.

10.3.3.5   ObjidValue

`ObjidValue` is mapped to the following complex type:

```
<xsd:complexType name="ObjidValue">
    <xsd:simpleContent>
       <xsd:extension base="SimpleTypes:TString">
             <xsd:attributeGroup ref="Values:ValueAtts"/>
       </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

-     `value`   The objid value as string.

-     `null`   If no value is given.

-     `omit`   If the value is omitted.

Attributes:

-     The same attributes as those of Value.

10.3.3.6　VerdictValue

VerdictValue is mapped to the following complex type:

```
<xsd:complexType name="VerdictValue">
    <xsd:simpleContent>
        <xsd:extension base="SimpleTypes:TString">
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

- value　The verdict value as string.
- null　If no value is given.
- omit　If the value is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.7　BitstringValue

BitstringValue is mapped to the following complex type:

```
<xsd:complexType name="BitstringValue">
    <xsd:simpleContent>
        <xsd:extension base="SimpleTypes:TString">
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

- value　The bitstring value as string.
- null　If no value is given.
- omit　If the value is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.8　HexstringValue

HexstringValue is mapped to the following complex type:

```
<xsd:complexType name="HexstringValue">
    <xsd:simpleContent>
        <xsd:extension base="SimpleTypes:TString">
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

- value　The hexstring value as string.
- null　If no value is given.
- omit　If the value is omitted.

Attributes:

The same attributes as those of Value.

10.3.3.9   OctetstringValue

OctetstringValue is mapped to the following complex type:

```
<xsd:complexType name="OctetstringValue">
   <xsd:simpleContent>
      <xsd:extension base="SimpleTypes:TString">
         <xsd:attributeGroup ref="Values:ValueAtts"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

*   `value`   The octetstring value as string.
*   `null`   If no value is given.
*   `omit`   If the value is omitted.

Attributes:

*   The same attributes as those of Value.

10.3.3.10   CharstringValue

CharstringValue is mapped to the following complex type:

```
<xsd:complexType name="CharstringValue">
   <xsd:simpleContent>
      <xsd:extension base="SimpleTypes:TString">
         <xsd:attributeGroup ref="Values:ValueAtts"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

*   `value`   The charstring value as string.
*   `null`   If no value is given.
*   `omit`   If the value is omitted.

Attributes:

*   The same attributes as those of Value.

10.3.3.11   UniversalCharstringValue

UniversalCharstringValue is mapped to the following complex type:

```
<xsd:complexType name="UniversalCharstringValue">
   <xsd:simpleContent>
      <xsd:extension base="SimpleTypes:TString">
         <xsd:attributeGroup ref="Values:ValueAtts"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

Simple content:

-    `value`   The universal charstring value as string.

-    `null`   If no value is given.

-    `omit`   If the value is omitted.

Attributes:

-    The same attributes as those of Value.

10.3.3.12 RecordValue

`RecordValue` is mapped to the following complex type:

```
<xsd:complexType name="RecordValue">
    <xsd:choice>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"/>
            <xsd:element name="charstring" type="Values:CharstringValue"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
        </xsd:choice>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd: choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Sequence of elements:

-    `integer`   An integer value.

-    `float`   A float value.

-    `boolean`   A boolean value.

-    `objid`   An objid value.

-    `verdicttype`   A verdicttype value.

-    `bitstring`   A bitstring value.

-    `hexstring`   An hexstring value.

-    `octetstring`   An octetstring value.

-    `charstring`   A charstring value.

-    `universal_charstring`   A universal charstring value.

-    `record`   A record value.

-    `record_of`   A record of value.

-    `set`   A set value.

-    `set_of`   A set of value.

-    `enumerated`   An enumerated value.

- union   A union value.
- anytype   An anytype value.
- address   An address value.
- null   If no field is given.
- omit   If the field is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.13 RecordOfValue

RecordOfValue is mapped to the following complex type:

```
<xsd:complexType name="RecordOfValue">
    <xsd:choice>
        <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of sequence of elements:

- integer   An integer value.
- float   A float value.
- boolean   A boolean value.
- objid   An objid value.
- verdicttype   A verdicttype value.
- bitstring   A bitstring value.
- hexstring   An hexstring value.

- • octetstring   An octetstring value.
- • charstring   A charstring value.
- • universal_charstring   A universal charstring value.
- • record   A record value.
- • record_of   A record of value.
- • set   A set value.
- • set_of   A set of value.
- • enumerated   An enumerated value.
- • union   A union value.
- • anytype   An anytype value.
- • address   An address value.
- • null   If no field is given.
- • omit   If the field is omitted.

Attributes:

- • The same attributes as those of Value.

10.3.3.14 SetValue

SetValue is mapped to the following complex type:

```
<xsd:complexType name="SetValue">
    <xsd:choice>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"/>
            <xsd:element name="charstring" type="Values:CharstringValue"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
        </xsd:choice>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Sequence of elements:

- • integer   An integer value.
- • float   A float value.
- • boolean   A boolean value.
- • objid   An objid value.
- • verdicttype   A verdicttype value.
- • bitstring   A bitstring value.
- • hexstring   An hexstring value.

- octetstring   An octetstring value.
- charstring   A charstring value.
- universal_charstring A universal charstring value.
- record   A record value.
- record_of   A record of value.
- set   A set value.
- set_of   A set of value.
- enumerated   An enumerated value.
- union   A union value.
- anytype   An anytype value.
- address   An address value.
- null   If no field is given.
- omit   If the field is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.15 SetOfValue

`SetOfValue` is mapped to the following complex type:

```
<xsd:complexType name="SetOfValue">
    <xsd:choice>
        <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd: element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="charstring" type="Values:CharstringValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="universal_charstring"
            type="Values:UniversalCharstringValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="set" type="Values:SetValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="set_of" type="Values:SetOfValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of sequence of elements:

- `integer` An integer value.
- `float` A float value.
- `boolean` A boolean value.
- `objid` An objid value.
- `verdicttype` A verdicttype value.
- `bitstring` A bitstring value.
- `hexstring` An hexstring value.
- `octetstring` An octetstring value.
- `charstring` A charstring value.
- `universal_charstring` A universal charstring value.
- `record` A record value.
- `record_of` A record of value.
- `set` A set value.
- `set_of` A set of value.
- `enumerated` An enumerated value.
- `union` A union value.
- `anytype` An anytype value.
- `address` An address value.
- `null` If no field is given.
- `omit` If the field is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.16 EnumeratedValue

`EnumeratedValue` is mapped to the following complex type:

```
<xsd:complexType name="EnumeratedValue">
    <xsd:choice>
        <xsd:element name="value" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Sequence of elements:

- `value` The enumeration value.
- `null` If no value is given.
- `omit` If the value is omitted.

Attributes:

- The same attributes as those of Value.

10.3.3.17 UnionValue

`UnionValue` is mapped to the following complex type:

```
<xsd:complexType name="UnionValue">
    <xsd:choice>
        <xsd:element name="integer" type="Values:IntegerValue"/>
        <xsd:element name="float" type="Values:FloatValue"/>
        <xsd:element name="boolean" type="Values:BooleanValue"/>
        <xsd:element name="objid" type="Values:ObjidValue"/>
        <xsd:element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:CharstringValue"/>
        <xsd:element name="universal_charstring"
            type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
        <xsd:element name="anytype" type="Values:AnytypeValue"/>
        <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of elements:

- `integer`   An integer value.

- `float`   A float value.

- `boolean`   A boolean value.

- `objid`   An objid value.

- `verdicttype`   A verdicttype value.

- `bitstring`   A bitstring value.

- `hexstring`   An hexstring value.

- `octetstring`   An octetstring value.

- `charstring`   A charstring value.

- `universal_charstring`   A universal charstring value.

- `record`   A record value.

- `record_of`   A record of value.

- `set`   A set value.

- `set_of`   A set of value.

- `enumerated`   An enumerated value.

- `union`   A union value.

- `anytype`   An anytype value.

- `address`   An address value.

Attributes:

- The same attributes as those of Value.

10.3.3.18 AnytypeValue

AnytypeValue is mapped to the following complex type:

```
<xsd:complexType name="AnytypeValue">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="integer" type="Values:IntegerValue"/>
        <xsd:element name="float" type="Values:FloatValue"/>
        <xsd:element name="boolean" type="Values:BooleanValue"/>
        <xsd:element name="objid" type="Values:ObjidValue"/>
        <xsd:element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:OctetstringValue"/>
        <xsd:element name="universal_charstring"
            type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
        <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of elements:

- `integer` An integer value.

- `float` A float value.

- `boolean` A boolean value.

- `objid` An objid value.

- `verdicttype` A verdicttype value.

- `bitstring` A bitstring value.

- `hexstring` An hexstring value.

- `octetstring` An octetstring value.

- `charstring` A charstring value.

- `universal_charstring` A universal charstring value.

- `record` A record value.

- `record_of` A record of value.

- `set` A set value.

- `set_of` A set of value.

- `enumerated` An enumerated value.

- `union` A union value.

- `address` An address value.

Attributes:

- The same attributes as those of Value.

10.3.3.19 AddressValue

AddressValue is mapped to the following complex type:

```
<xsd:complexType name="AddressValue">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="integer" type="Values:IntegerValue"/>
```

```
        <xsd:element name="float" type="Values:FloatValue"/>
        <xsd:element name="boolean" type="Values:BooleanValue"/>
        <xsd:element name="objid" type="Values:ObjidValue"/>
        <xsd:element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:OctetstringValue"/>
        <xsd:element name="universal_charstring"
            type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
        <xsd:element name="anytype" type="Values:AnytypeValue"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
 </xsd:complexType>
```

Choice of elements:

-      `integer`   An integer value.

-      `float`   A float value.

-      `boolean`   A boolean value.

-      `objid`   An objid value.

-      `verdicttype`   A verdicttype value.

-      `bitstring`   A bitstring value.

-      `hexstring`   An hexstring value.

-      `octetstring`   An octetstring value.

-      `charstring`   A charstring value.

-      `universal_charstring`   A universal charstring value.

-      `record`   A record value.

-      `record_of`   A record of value.

-      `set`   A set value.

-      `set_of`   A set of value.

-      `enumerated`   An enumerated value.

-      `union`   A union value.

-      `anytype`   An anytype value.

Attributes:

-      The same attributes as those of Value.

10.3.4 Abstract logging types mapping

Additional types are defined to ease the logging of matches between values and templates.

10.3.4.1 TciValueTemplate

TciValueTemplate is mapped to the following complex type:

```
    <xsd:complexType name="TciValueTemplate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Values:Value">
            <xsd:choice>
```

```
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name= "verdicttype" type="values:verdictValue"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                     type="Templates:UniversalCharstringTemplate"/>
                    <xsd:element name="record" type="Templates:RecordTemplate"/>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                    <xsd:element name="set" type="Templates:SetTemplate"/>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                    <xsd:element name="union" type="Templates:UnionTemplate"/>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                    <xsd:element name="address" type="Templates:AddressTemplate"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

Choice of elements:

- integer   An integer template.
- float   A float template.
- boolean   A boolean template.
- objid   An objid template.
- verdicttype   A verdicttype template.
- bitstring   A bitstring template.
- hexstring   An hexstring template.
- octetstring   An octetstring template.
- charstring   A charstring template.
- universal_charstring   A universal charstring template.
- record   A record template.
- record_of   A record of template.
- set   A set template.
- set_of   A set of template.
- enumerated   An enumerated template.
- union   A union template.
- anytype   An anytype template.
- address   An address template.
- omit   An omit template.
- any   An any template.
- anyoromit   An anyoromit template.
- templateDef   A complex template definition.

Attributes:

- none.

10.3.4.2 TciNonValueTemplate

TciNonValueTemplate is mapped to the following complex type:

```
<xsd:complexType name="TciNonValueTemplate">
    <xsd:sequence>
      <xsd:choice>
          <xsd:element name="any" type="Templates:any"/>
          <xsd:element name="all" type="Templates:all"/>
          <xsd:element name="templateDef" type="SimpleTypes:TString"/>
          <xsd:element name="null" type="Templates:null"/>
      </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

Choice of elements:

- `any`   An any template.

- `all`   An all template.

- `templateDef`   A complex template definition.

- `null`   No template is given.

Attributes:

- none.

10.3.4.3 TciValueList

TciValueList is mapped to the following complex type:

```
<xsd:complexType name="TciValueList">
    <xsd:sequence>
      <xsd:element name="val" type="Values:Value"
       maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

Sequence of elements:

- `val`   The values in the value list.

Attributes:

- none.

10.3.4.4 TciValueDifference

TciValueDifference is mapped to the following complex type:

```
<xsd:complexType name="TciValueDifference">
     <xsd:sequence>
          <xsd:element name="val" type="SimpleTypes:xpath"/>
          <xsd:element name="tmpl" type="SimpleTypes:xpath"/>
     </xsd:sequence>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
    <xsd:attribute name="desc" type="SimpleTypes:TString" use="optional"/>
</xsd:complexType>
```

Sequence of elements:

- `val`   A reference to the mismatching value.

- `tmpl`   A reference to the template.

Attributes:

- The same attributes as those of Value.

- `desc`   The reason of the mismatch.

10.3.4.5   TciValueDifferenceList

`TciValueDifferenceList` is mapped to the following complex type:

```
<xsd:complexType name="TciValueDifferenceList">
   <xsd:sequence>
      <xsd:element name="diff" type="Templates:TciValueDifference"
       maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

Sequence of elements:

- `diff`   The value/template differences in the value difference list.

Attributes:

- none.

10.4   Mapping of the operations on the logging interface

Every operation provided at the logging interface has a corresponding complex type definition in XML. These complex type definitions are extensions of Event.

10.4.1   Event

`Event` is mapped to the following complex type:

```
<!-- common definition for all events -->
<xsd:complexType name="Event" mixed="true">
   <xsd:sequence>
      <xsd:element name="am" type="SimpleTypes:TString"/>
   </xsd:sequence>
   <xsd:attribute name="ts" type="xsd:long" use="required"/>
   <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
   <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

<!-- general identifier structure for test components, ports and timer -->
   <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
   <xsd:attribute name="id" type="SimpleTypes:TString " use="required"/>
   <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

- `am`   A message, to be used for further information in the log.

Attributes:

- `ts`   The time when the event is produced.

- `src`   The source file of the test specification.

- `line`   The line number where the request is performed.

- `name`   The name of the component which produces this event.

- `id`   The id of the component which produces this event.

- `type`   The type of the component which produces this event.

10.4.2   Mapping of operations

The mapping for the operations is given in the following.

TCI-TL provided	

void tliTcExecute
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TciTestCaseIdType tcId,
 	
in TriParameterListType triPars,
 	
in TriTimerDurationType dur)
	`<xsd:complexType name="tliTcExecute">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="tcId"`
	`type="Types:TciTestCaseIdType"/>`
	`<xsd:element name="triPars"`
	`type="Types:TriParameterListType" minOccurs="0"/>`
	`<xsd:element name="dur"`
	`type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTcStart
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TciTestCaseIdType tcId,
 	
in TciParameterListType tciPars,
 	
in TriTimerDurationType dur)
	`<xsd:complexType name="tliTcStart">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="tcId"`
	`type="Types:TciTestCaseIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="dur"`
	`type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTcStop(in TString am, in TInteger ts,
 	
in TString src, in TInteger line,
 	
in TriComponentIdType c)
	`<xsd:complexType name="tliTcStop">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTcStarted
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TciTestCaseIdType tcId,
 	
in TciParameterListType tciPars,
 	
in TriTimerDurationType dur)
	`<xsd:complexType name="tliTcStarted">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="tcId"`
	`type="Types:TciTestCaseIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="dur"`
	`type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliTcTerminated
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line,
 	
 in TriComponentIdType c,
 	
 in TciTestCaseIdType tcId,
 	
 in TciParameterListType tciPars,
 	
 in VerdictValue verdict)
	`<xsd:complexType name="tliTcTerminated">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="tcId"`
	`type="Types:TciTestCaseIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="verdict"`
	`type="Values:VerdictValue"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCtrlStart
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line,
 	
 in TriComponentIdType c)
	`<xsd:complexType name="tliCtrlStart">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCtrlStop
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c)
	`<xsd:complexType name="tliCtrlStop">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCtrlTerminated
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c)
	`<xsd:complexType name="tliCtrlTerminated">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMSend_m
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortId to, in Value msgValue,
 	
 in TriAddressType address,
 	
 in TciStatusType encoderFailure,
 	
 in TriMessageType msg,
 	
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliMSend_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="msgValue" type="Values:Value"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="msg" type="Types:TriMessageType"`
	`minOccurs="0"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliMSend_m_BC
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType to,
 	
 in Value msgValue,
 	
 in TciStatusType encoderFailure,
 	
 in TriMessageType msg,
 	
 in TriStatusType transmissionFailure)
	``xsd
	<xsd:complexType name="tliMSend_m_BC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="msgValue" type="Values:Value"/>
	<xsd:choice>
	<xsd:element name="encoder-failure"
	type="SimpleTypes:TciStatusType" minOccurs="0"/>
	<xsd:sequence>
	<xsd:element name="msg" type="Types:TriMessageType"
	minOccurs="0"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:choice>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliMSend_m_MC
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdListType to,
in Value msgValue,
 	
 in TriAddressListType addresses,
 	
 in TciStatusType encoderFailure,
 	
 in TriMessageType msg,
 	
 in TriStatusType transmissionFailure)
	``xsd
	<xsd:complexType name="tliMSend_m_MC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="msgValue" type="Values:Value"/>
	<xsd:element name="addresses"
	type="Types:TriAddressListType" minOccurs="0"/>
	<xsd:choice>
	<xsd:element name="encoder-failure"
	type="SimpleTypes:TciStatusType" minOccurs="0"/>
	<xsd:sequence>
	<xsd:element name="msg" type="Types:TriMessageType"
	minOccurs="0"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:choice>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliMSend_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
 	
in Value msgValue,
 	
in TriStatusType transmissionFailure)
	``xsd
	<xsd:complexType name="tliMSend_c">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
	<xsd:element name="msgValue" type="Values:Value"/>
	<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliMSend_c_BC
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdListType to,
 	
in Value msgValue,
 	
in TriStatusType transmissionFailure)
	``xsd
	<xsd:complexType name="tliMSend_c_BC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
	<xsd:element name="msgValue" type="Values:Value"/>
	<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliMSend_c_MC
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdListType to,
 	
in Value msgValue,
 	
in TriStatusType transmissionFailure)
	``xsd
	<xsd:complexType name="tliMSend_c_MC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
	<xsd:element name="msgValue" type="Values:Value"/>
	<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliMDetected_m
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType from,
in TriMessageType msg,
 in TriAddressType address)
	`<xsd:complexType name="tliMDetected_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="msgValue" type="Types:TriMessageType"/>`
	`<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMDetected_c
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType from,
 in Value msgValue)
	`<xsd:complexType name="tliMDetected_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="msgValue" type="Values:Value"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMMismatch_m
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in Value msgValue,
 in TciValueTemplate msgTmpl,
 in TciValueDifferenceList diffs,
 in TriAddressType address,
 in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliMMismatch_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="msgValue" type="Values:Value"/>`
	`<xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>`
	`<xsd:element name="addressTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliMMismatch_c
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in Value msgValue,
 in TciValueTemplate msgTmpl,
 in TciValueDifferenceList diffs,
 in TriComponentIdType from,
 in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliMMismatch_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="msgValue" type="Values:Value"/>`
	`<xsd:element name="msgTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="from"`
	`type="Types:TriComponentIdType" minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciNonValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMReceive_m
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in Value msgValue,
 in TciValueTemplate msgTmpl,
 in TriAddressType address,
 in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliMReceive_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="msgValue" type="Values:Value"`
	`minOccurs="0"/>`
	`<xsd:element name="msgTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:element name="addressTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMReceive_c
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in Value msgValue,
 in TciValueTemplate msgTmpl,
 in TriComponentIdType fromComp,
 in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliMReceive_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="msgValue" type="Values:Value"`
	`minOccurs="0"/>`
	`<xsd:element name="msgTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`<xsd:element name="fromComp"`
	`type="Types:TriComponentIdType" minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciNonValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrCall_m
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TriAddressType address,
 in TciStatusType encoderFailure,
 in TriParameterListType triPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="triPars"`
	`type="Types:TriParameterListType"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCall_m_BC
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TciStatusType encoderFailure,
 in TriParameterListType triPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_m_BC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="triPars"`
	`type="Types:TriParameterListType"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrCall_m_MC
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TriAddressListType addresses,
 in TciStatusType encoderFailure,
 in TriParameterListType triPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_m_MC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="triPars" type="Types:TriParameterListType"/>`
	`<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCall_c
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrCall_c_BC
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdListType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_c_BC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCall_c_MC
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdListType to,
 in TriSignatureIdType signature,
 in TciParameterListType tciPars,
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrCall_c_MC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>`
	`<xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrGetCallDetected_m
 	
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at, in TriPortIdType from,
 in TriSignatureIdType signature,
 in TriParameterListType triPars,
 in TriAddressType address)
	`<xsd:complexType name="tliPrGetCallDetected_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="triPars" type="Types:TriParameterListType"/>`
	`<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrGetCallDetected_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType from,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars)
	`<xsd:complexType name="tliPrGetCallDetected_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrGetCallMismatch_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in TciValueDifferenceList diffs,
 	
in TriAddressType address,
 	
in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliPrGetCallMismatch_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType"/>`
	`<xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>`
	`<xsd:element name="addressTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrGetCallMismatch_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in TciValueDifferenceList diffs,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliPrGetCallMismatch_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="parsTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="from"`
	`type="Types:TriComponentIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrGetCall_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in TriAddressType address,
 	
in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliPrGetCall_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="parsTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:element name="addressTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrGetCall_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciValueTemplate signatureTmpl
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliPrGetCall_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="signatureTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="parsTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="from"`
	`type="Types:TriComponentIdType" minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciNonValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrReply_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value replValue,
 	
in TriAddressType address,
 	
in TciStatusType encoderFailure,
 	
in TriParameterListType triPars,
 	
in TriParameterType repl,
 	
in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrReply_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="replValue" type="Values:Value"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="triPars"`
	`type="Types:TriParameterListType"/>`
	`<xsd:element name="repl" type="Types:TriParameterType"`
	`minOccurs="0"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrReply_m_BC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value replValue,
in TciStatusType encoderFailure,
in TriParameterListType triPars,
in TriParameterType repl,
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrReply_m_BC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:choice>
	<xsd:element name="encoder-failure"
	type="SimpleTypes:TciStatusType" minOccurs="0"/>
	<xsd:sequence>
	<xsd:element name="triPars"
	type="Types:TriParameterListType"/>
	<xsd:element name="repl" type="Types:TriParameterType"
	minOccurs="0"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:choice>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrReply_m_MC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value replValue,
in TriAddressListType addresses,
in TciStatusType encoderFailure,
in TriParameterListType triPars,
in TriParameterType repl,
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrReply_m_MC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:element name="addresses"
	type="Types:TriAddressListType" minOccurs="0"/>
	<xsd:choice>
	<xsd:element name="encoder-failure"
	type="SimpleTypes:TciStatusType" minOccurs="0"/>
	<xsd:sequence>
	<xsd:element name="triPars"
	type="Types:TriParameterListType"/>
	<xsd:element name="repl" type="Types:TriParameterType"
	minOccurs="0"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:choice>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliPrReply_c
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value replValue,
in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrReply_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="replValue" type="Values:Value"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrReply_c_BC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdListType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value replValue,
in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrReply_c_BC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdListType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="replValue" type="Values:Value"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrReply_c_MC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdListType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value replValue,
in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrReply_c_MC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdListType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="replValue" type="Values:Value"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrGetReplyDetected_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType from,
 	
in TriSignatureIdType signature,
 	
in TriParameterListType triPars,
 	
in TriParameterType repl,
 	
in TriAddressType address)
	`<xsd:complexType name="tliPrGetReplyDetected_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="triPars" type="Types:TriParameterListType"/>`
	`<xsd:element name="repl" type=" Types:TriParameterType"/>`
	`<xsd:element name="address" type="Types:TriAddressType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrGetReplyDetected_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType from,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value replValue)
	`<xsd:complexType name="tliPrGetReplyDetected_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>`
	`<xsd:element name="signature" type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars" type="Types:TciParameterListType"/>`
	`<xsd:element name="replValue" type="Values:Value"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrGetReplyMismatch_m
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in TciValueTemplate parsTmpl,
in Value replValue,
in TciValueTemplate replyTmpl,
in TciValueDifferenceList diffs,
in TriAddressType address,
in TciValueTemplate addressTmpl)
	``xml
	<xsd:complexType name="tliPrGetReplyMismatch_m">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="parsTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:element name="replTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="diffs"
	type="Templates:TciValueDifferenceList"/>
	<xsd:element name="address" type="Types:TriAddressType"
	minOccurs="0"/>
	<xsd:element name="addressTmpl"
	type="Templates:TciValueTemplate" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrGetReplyMismatch_c
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in TciValueTemplate parsTmpl,
in Value replValue,
in TciValueTemplate replyTmpl,
in TciValueDifferenceList diffs,
in TriComponentIdType from,
in TciNonValueTemplate fromTmpl)
	``xml
	<xsd:complexType name="tliPrGetReplyMismatch_c">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="parsTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:element name="replTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="diffs"
	type="Templates:TciValueDifferenceList"/>
	<xsd:element name="from"
	type="Types:TriComponentIdType" minOccurs="0"
	minOccurs="0"/>
	<xsd:element name="fromTmpl"
	type="Templates:TciNonValueTemplate" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliPrGetReply_m
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in Value replValue,
 	
in TciValueTemplate replyTmpl,
 	
in TriAddressType address,
 	
in TciValueTemplate addressTmpl)
	``xsd
	<xsd:complexType name="tliPrGetReply_m">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="parsTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:element name="replTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="address" type="Types:TriAddressType"
	minOccurs="0"/>
	<xsd:element name="addressTmpl"
	type="Templates:TciValueTemplate" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrGetReply_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in TciValueTemplate parsTmpl,
 	
in Value replValue,
 	
in TciValueTemplate replyTmpl,
 	
in TriComponentIdType from,
 	
in TciNonValueTemplate fromTmpl)
	``xsd
	<xsd:complexType name="tliPrGetReply_c">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="parsTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="replValue" type="Values:Value"/>
	<xsd:element name="replTmpl"
	type="Templates:TciValueTemplate"/>
	<xsd:element name="from"
	type="Types:TriComponentIdType" minOccurs="0"/>
	<xsd:element name="fromTmpl"
	type="Templates:TciNonValueTemplate" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliPrRaise_m
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType to,
 	
 in TriSignatureIdType signature,
 	
 in TciParameterListType tciPars,
 	
 in Value excValue, in TriAddressType
address,
 	
 in TciStatusType encoderFailure,
 	
 in TriExceptionType exc,
 	
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrRaise_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="exc" type="Types:TriExceptionType"`
	`minOccurs="0"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrRaise_m_BC
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType to,
 	
 in TriSignatureIdType signature,
 	
 in TciParameterListType tciPars,
 	
 in Value excValue,
 	
 in TciStatusType encoderFailure,
 	
 in TriExceptionType exc,
 	
 in TriStatusType transmissionFailure)
	`<xsd:complexType name="tliPrRaise_m_BC">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="to" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:choice>`
	`<xsd:element name="encoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`<xsd:sequence>`
	`<xsd:element name="exc" type="Types:TriExceptionType"`
	`minOccurs="0"/>`
	`<xsd:element name="transmission-failure"`
	`type="SimpleTypes:TriStatusType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:choice>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrRaise_m_MC
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value excValue,
 	
in TriAddressListType addresses,
 	
in TciStatusType encoderFailure,
 	
in TriExceptionType exc,
 	
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrRaise_m_MC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="excValue" type="Values:Value"/>
	<xsd:element name="addresses"
	type="Types:TriAddressListType" minOccurs="0"/>
	<xsd:choice>
	<xsd:element name="encoder-failure"
	type="SimpleTypes:TciStatusType" minOccurs="0"/>
	<xsd:sequence>
	<xsd:element name="exc" type="Types:TriExceptionType"
	minOccurs="0"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:choice>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrRaise_c
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TriPortIdType at, in TriPortIdType to,
 	
in TriSignatureIdType signature,
 	
in TciParameterListType tciPars,
 	
in Value excValue,
 	
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrRaise_c">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="excValue" type="Values:Value"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliPrRaise_c_BC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdListType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value excValue,
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrRaise_c_BC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdListType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="excValue" type="Values:Value"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrRaise_c_MC
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdListType to,
in TriSignatureIdType signature,
in TciParameterListType tciPars,
in Value excValue,
in TriStatusType transmissionFailure)
	``xml
	<xsd:complexType name="tliPrRaise_c_MC">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="to" type="Types:TriPortIdListType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="tciPars"
	type="Types:TciParameterListType"/>
	<xsd:element name="excValue" type="Values:Value"/>
	<xsd:element name="transmission-failure"
	type="SimpleTypes:TriStatusType" minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliPrCatchDetected_m
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at, in TriPortIdType from,
in TriSignatureIdType signature,
in TriExceptionType exc,
in TriAddressType address)
	``xml
	<xsd:complexType name="tliPrCatchDetected_m">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="at" type="Types:TriPortIdType"/>
	<xsd:element name="from" type="Types:TriPortIdType"
	minOccurs="0"/>
	<xsd:element name="signature"
	type="Types:TriSignatureIdType"/>
	<xsd:element name="exc" type="Types:TriExceptionType"/>
	<xsd:element name="address" type="Types:TriAddressType"
	minOccurs="0"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliPrCatchDetected_c
 	
 (in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at, in TriPortIdType from,
 	
 in TriSignatureIdType signature,
 	
 in Value excValue)
	`<xsd:complexType name="tliPrCatchDetected_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="from" type="Types:TriPortIdType"`
	`minOccurs="0"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCatchMismatch_m
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at,
 	
 in TriSignatureIdType signature,
 	
 in Value excValue,
 	
 in TciValueTemplate excTmpl,
 	
 in TciValueDifferenceList diffs,
 	
 in TriAddressType address,
 	
 in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliPrCatchMismatch_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:element name="excTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="address"`
	`type="Types:TriAddressType"/>`
	`<xsd:element name="addressTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCatchMismatch_c
 	
(in TString am, in TInteger ts, in TString src,
 	
 in TInteger line, in TriComponentIdType c,
 	
 in TriPortIdType at,
 	
 in TriSignatureIdType signature,
 	
 in Value excValue,
 	
 in TciValueTemplate excTmpl,
 	
 in TciValueDifferenceList diffs,
 	
 in TriComponentIdType from,
 	
 in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliPrCatchMismatch_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:element name="excTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`<xsd:element name="from"`
	`type="Types:TriComponentIdType" minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciNonValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrCatch_m
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at,
 in TriSignatureIdType signature,
 in Value excValue,
 in TciValueTemplate excTmpl,
 in TriAddressType address,
 in TciValueTemplate addressTmpl)
	`<xsd:complexType name="tliPrCatch_m">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:element name="excTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="address" type="Types:TriAddressType"`
	`minOccurs="0"/>`
	`<xsd:element name="addressTmpl"`
	`type="Templates:TciValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCatch_c
 	
 (in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TriPortIdType at,
 in TriSignatureIdType signature,
 in Value excValue,
 in TciValueTemplate excTmpl,
 in TriComponentIdType from,
 in TciNonValueTemplate fromTmpl)
	`<xsd:complexType name="tliPrCatch_c">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`<xsd:element name="excValue" type="Values:Value"/>`
	`<xsd:element name="excTmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="from"`
	`type="Types:TriComponentIdType" minOccurs="0"/>`
	`<xsd:element name="fromTmpl"`
	`type="Templates:TciNonValueTemplate" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature)	`<xsd:complexType name="tliPrCatchTimeoutDetected">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliPrCatchTimeout
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriPortIdType at,
in TriSignatureIdType signature)
	`<xsd:complexType name="tliPrCatchTimeout">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="at" type="Types:TriPortIdType"/>`
	`<xsd:element name="signature"`
	`type="Types:TriSignatureIdType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCCreate
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriComponentIdType comp,
in TString name, in TBoolean alive)
	`<xsd:complexType name="tliCCreate">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp"`
	`type="Types:TriComponentIdType"/>`
	`<xsd:element name="name" type="SimpleTypes:TString"/>`
	`<xsd:element name="alive" type="SimpleTypes:TBoolean"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCStart
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriComponentIdType comp,
in TciBehaviourIdType name,
in TciParameterListType tciPars)
	`<xsd:complexType name="tliCStart">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp"`
	`type="Types:TriComponentIdType"/>`
	`<xsd:element name="name"`
	`type="Types:TciBehaviourIdType"/>`
	`<xsd:element name="tciPars"`
	`type="Types:TciParameterListType" minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCRunning
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriComponentIdType comp,
in ComponentStatusType status)
	`<xsd:complexType name="tliCRunning">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp"`
	`type="Types:TriComponentIdType"/>`
	`<xsd:element name="status"`
	`type="SimpleTypes:ComponentStatusType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliCAlive (in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in ComponentStatusType status)	`<xsd:complexType name="tliCAlive">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp" type="Types:TriComponentIdType"/>`
	`<xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	`<xsd:complexType name="tliCStop">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp" type="Types:TriComponentIdType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCKill (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	`<xsd:complexType name="tliCKill">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp" type="Types:TriComponentIdType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCDoneMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)	`<xsd:complexType name="tliCDoneMismatch">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="comp" type="Types:TriComponentIdType"/>`
	`<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliCDone (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	`<xsd:complexType name="tliCDone">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	
void tliCKilledMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)	``` <xsd:complexType name="tliCKilledMismatch"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliCKilled (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	``` <xsd:complexType name="tliCKilled"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliCTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	``` <xsd:complexType name="tliCTerminated"> without verdict) --> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="verdict" type="Values:VerdictValue" /> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliPConnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2)	``` <xsd:complexType name="tliPConnect"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType> ```
void tliPDisconnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	``` <xsd:complexType name="tliPDisconnect"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType> ```

TCI-TL provided	
void tliPMap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2)	``` <xsd:complexType name="tliPMap"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType> ```
void tliPUnmap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	``` <xsd:complexType name="tliPUnmap"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType> ```
void tliPClear (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	``` <xsd:complexType name="tliPClear"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortStatus"/> </xsd:complexContent> </xsd:complexType> ```
void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	``` <xsd:complexType name="tliPStart"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortStatus"/> </xsd:complexContent> </xsd:complexType> ```
void tliPStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	``` <xsd:complexType name="tliPStop"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortStatus"/> </xsd:complexContent> </xsd:complexType> ```
void tliPHalt (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	``` <xsd:complexType name="tliPHalt"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortStatus"/> </xsd:complexContent> </xsd:complexType> ```
void tliEncode (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TciStatusType encoderFailure, in TriMessageType msg, in TString codec)	``` <xsd:complexType name="tliEncode"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="val" type="Values:Value"/> <xsd:choice> <xsd:element name="msg" type="Types:TriMessageType"/> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType" minOccurs="0"/> </xsd:choice> <xsd:element name="codec" type="SimpleTypes:TString" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```

TCI-TL provided	

void tliDecode
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriMessageType msg,
in TciStatusType decoderFailure, in Value val,
in TString codec)
	`<xsd:complexType name="tliDecode" mixed="true">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:choice>`
	`<xsd:element name="val" type="Values:Value"/>`
	`<xsd:element name="decoder-failure"`
	`type="SimpleTypes:TciStatusType" minOccurs="0"/>`
	`</xsd:choice>`
	`<xsd:element name="msg" type="Types:TriMessageType"/>`
	`<xsd:element name="codec" type="SimpleTypes:TString"`
	` minOccurs="0"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTTimeoutDetected
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer)
	`<xsd:complexType name="tliTTimeoutDetected">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType" />`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTTimeoutMismatch
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TciNonValueTemplate timerTmpl)
	`<xsd:complexType name="tliTTimeoutMismatch">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType" />`
	`<xsd:element name="timerTmpl"`
	`type="Templates:TciNonValueTemplate"  />`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTTimeout
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TciNonValueTemplate timerTmpl)
	`<xsd:complexType name="tliTTimeout">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType" />`
	`<xsd:element name="timerTmpl"`
	`type="Templates:TciNonValueTemplate"  />`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliTStart
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TriTimerDurationType dur)
	`<xsd:complexType name="tliTStart">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType"/>`
	`<xsd:element name="dur"`
	`type="SimpleTypes:TriTimerDurationType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTStop
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TriTimerDurationType dur)
	`<xsd:complexType name="tliTStop">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType"/>`
	`<xsd:element name="dur"`
	`type="SimpleTypes:TriTimerDurationType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTRead
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TriTimerDurationType elapsed)
	`<xsd:complexType name="tliTRead">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType"/>`
	`<xsd:element name="elapsed"`
	`type="SimpleTypes:TriTimerDurationType"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliTRunning
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TriTimerIdType timer,
in TimerStatusType status)
	`<xsd:complexType name="tliTRunning">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="timer" type="Types:TriTimerIdType"/>`
	`</xsd:sequence>`
	`<xsd:attribute name="status"`
	`type="SimpleTypes:TimerStatusType"/>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	
void tliSEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars, in TString kind)	``` <xsd:complexType name="tliSEnter"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" /> <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/> <xsd:element name="kind" type="SimpleTypes:TString"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliSLeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars, in Value returnValue, in TString kind)	``` <xsd:complexType name="tliSLeave"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" /> <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/> <xsd:element name="returnValue" type="Values:Value" minOccurs="0"/> <xsd:element name="kind" type="SimpleTypes:TString"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliVar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value varValue)	``` <xsd:complexType name="tliVar"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" /> <xsd:element name="val" type="Values:Value" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```
void tliModulePar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in Value parValue)	``` <xsd:complexType name="tliModulePar"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" /> <xsd:element name="val" type="Values:Value" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> ```

TCI-TL provided	

void tliGetVerdict
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in VerdictValue verdict)
	`<xsd:complexType name="tliGetVerdict">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="verdict"`
	`type="Values:VerdictValue"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliSetVerdict
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in VerdictValue verdict)
	`<xsd:complexType name="tliSetVerdict">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="verdict"`
	`type="Values:VerdictValue"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliLog
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c,
in TString log)
	`<xsd:complexType name="tliLog">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="log" type="SimpleTypes:TString"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliAEnter
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c)
	`<xsd:complexType name="tliAEnter">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliALeave
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c)
	`<xsd:complexType name="tliALeave">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliADefaults
 	
(in TString am, in TInteger ts, in TString src,
in TInteger line, in TriComponentIdType c)
	`<xsd:complexType name="tliADefaults">`
	`<xsd:complexContent>`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

TCI-TL provided	

void tliAActivate
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in QualifiedName name,
 	
in TciParameterListType tciPars, in Value ref)
	``xml
	<xsd:complexType name="tliAActivate">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="name" type="Types:QualifiedName" />
	<xsd:element name="tciPars"
	type="Types:TciParameterListType" minOccurs="0"/>
	<xsd:element name="ref" type="Values:Value"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliADeactivate
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in Value ref)
	``xml
	<xsd:complexType name="tliADeactivate">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="ref" type="Values:Value"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliANomatch
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c)
	``xml
	<xsd:complexType name="tliANomatch">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event"/>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliARepeat
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c)
	``xml
	<xsd:complexType name="tliARepeat">
	<xsd:complexContent>
	<xsd:extension base="Events:Event"/>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliAWait
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c)
	``xml
	<xsd:complexType name="tliAWait">
	<xsd:complexContent>
	<xsd:extension base="Events:Event"/>
	</xsd:complexContent>
	</xsd:complexType>
	``

void tliAction
 	
(in TString am, in TInteger ts, in TString src,
 	
in TInteger line, in TriComponentIdType c,
 	
in TString action)
	``xml
	<xsd:complexType name="tliAction">
	<xsd:complexContent mixed="true">
	<xsd:extension base="Events:Event">
	<xsd:sequence>
	<xsd:element name="action" type="SimpleTypes:TString"/>
	</xsd:sequence>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	``

TCI-TL provided	

void tliMatch
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in Value expr, in TciValueTemplate tmpl)
	`<xsd:complexType name="tliMatch">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="expr" type="Values:Value"/>`
	`<xsd:element name="tmpl"`
	`type="Templates:TciValueTemplate"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliMatchMismatch
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in Value expr, in TciValueTemplate tmpl,
 in TciValueDifferenceList diffs)
	`<xsd:complexType name="tliMatchMismatch">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="expr" type="Values:Value"/>`
	`<xsd:element name="tmpl"`
	`type="Templates:TciValueTemplate"/>`
	`<xsd:element name="diffs"`
	`type="Templates:TciValueDifferenceList"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

void tliInfo
 	
(in TString am, in TInteger ts, in TString src,
 in TInteger line, in TriComponentIdType c,
 in TInteger level, in TString info)
	`<xsd:complexType name="tliInfo">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event">`
	`<xsd:sequence>`
	`<xsd:element name="level" type="SimpleTypes:TInteger"/>`
	`<xsd:element name="info" type="SimpleTypes:TString"/>`
	`</xsd:sequence>`
	`</xsd:extension>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

11   Use scenarios

This clause contains use scenarios that should help users of the TCI and tool vendors providing the TCI understand the semantics of the operations defined within this Recommendation.

The scenarios are defined in terms of UML sequence diagrams. The sequence diagram shows the interactions between the TCI entities. The scenarios are explained and where applicable underpinned with a TTCN-3 fragment corresponding to the scenario.

11.1   Initialization, collecting information, logging

11.1.1   Use scenario: initialization

The scenario in Figure 9 shows the initialization phase for a test system when a TTCN-3 module is to be selected for execution. At first, a root module has to be set with `tciRootModule`. The module parameters of the root module can be obtained with `tciGetModuleParameters`. Module parameter information can be used to ask the test system user for concrete values for each module parameter. The list of test cases available in the root module can be retrieved with `tciGetTestCases`. These test cases can be directly executed from the test management. Their parameters and their test system interface can be obtained with `tciGetTestCaseParameters` and `tciGetTestCaseTSI`,

respectively.

11.1.1.1   Sequence diagram
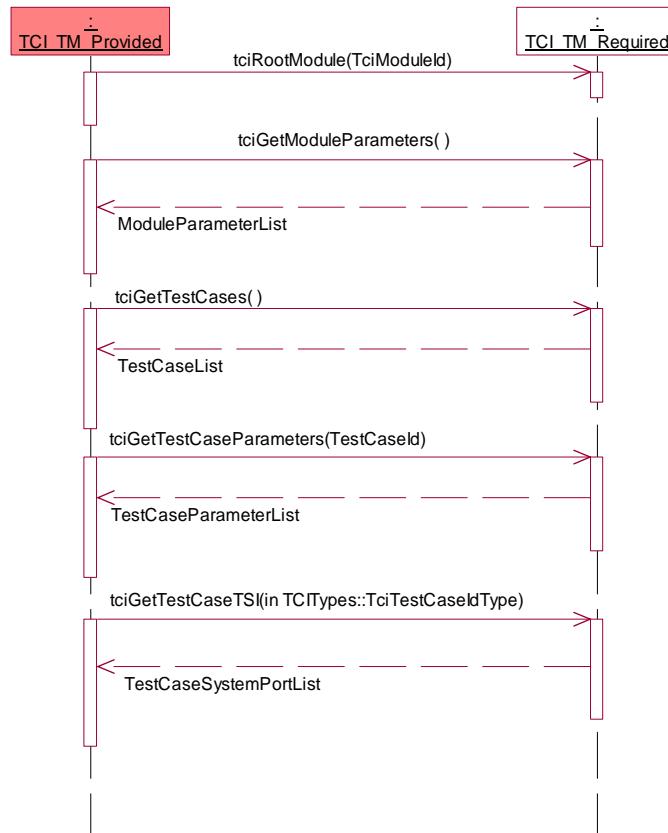
Figure 9 – Use scenario – initialization

11.1.1.2 TTCN-3 fragment

The initialization is outside the scope of TTCN-3.

11.1.2 Use scenario: requesting module parameters

The scenario in Figure 10 shows how a test component requests the actual value of a module parameter needed for the execution of its test behaviour. At first, the type of a module parameter is requested, then the value can be constructed by the TM and given to the TE.

11.1.2.1 Sequence diagram
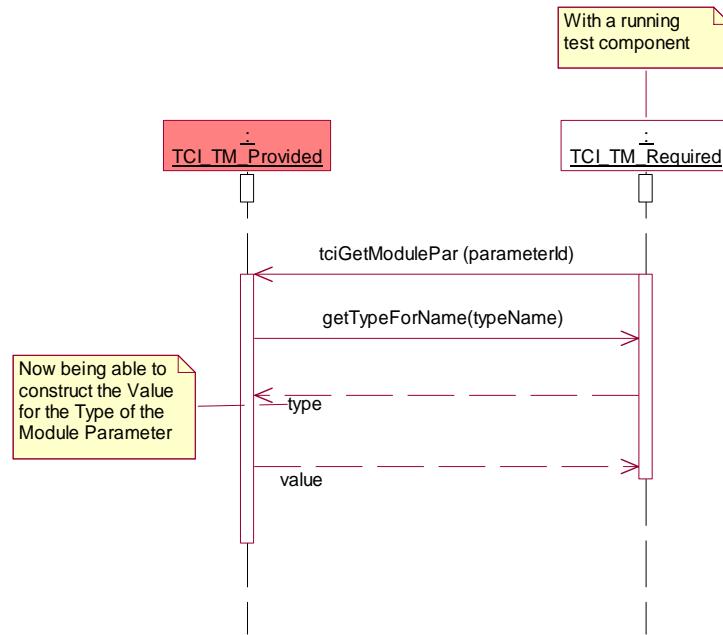
Figure 10 – Use scenario – requesting module Pars

11.1.2.2 TTCN-3 fragment

```
module AModule {
    …
    modulepar {
        integer AModulePar
    }
    …
    function AFunction (…) … {
        integer x;
        …
        x:= 2+AModulePar; // an expression with a module parameter
        …
    }
    …
}
```

11.1.3 Use scenario: logging

The scenario in Figure 11 shows logging of information during the execution of a test behaviour by a test component. The message to be logged is propagated to the test logging.

11.1.3.1 Sequence diagram
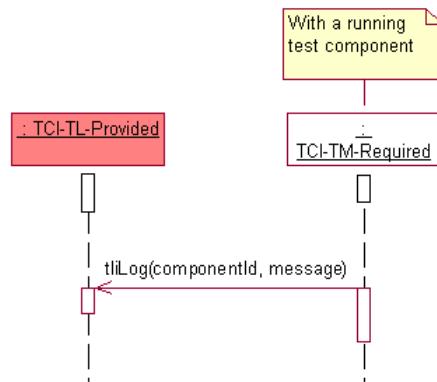
Figure 11 – Use scenario – logging

11.1.3.2　TTCN-3 fragment

```
module AModule {
    …
    function AFunction (…) … {
        …
        log('AMessage');
        …
    }
    …
}
```

11.2　Execution of test cases and control

11.2.1　Use scenario: execution of control

The scenario in Figure 12 shows the sequence of operations to execute the control part of a TTCN-3 module. The module containing the control part is selected first, then the control is started, then it is executed until the execution is terminated by TE.

11.2.1.1　Sequence diagram
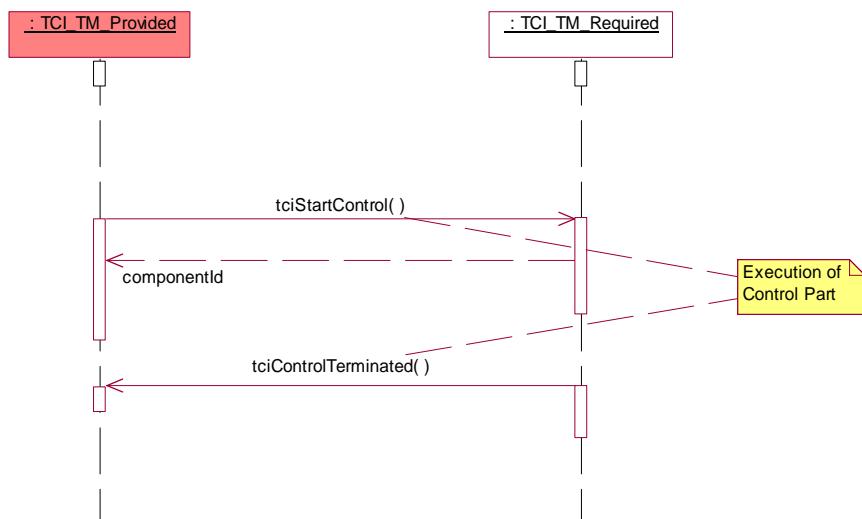
Figure 12 – Use scenario – execution of control

11.2.1.2   TTCN-3 fragment

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.2.2   Use scenario: test case execution within control

The scenario in Figure 13 shows how a test case is executed within the control part.

11.2.2.1   Sequence diagram
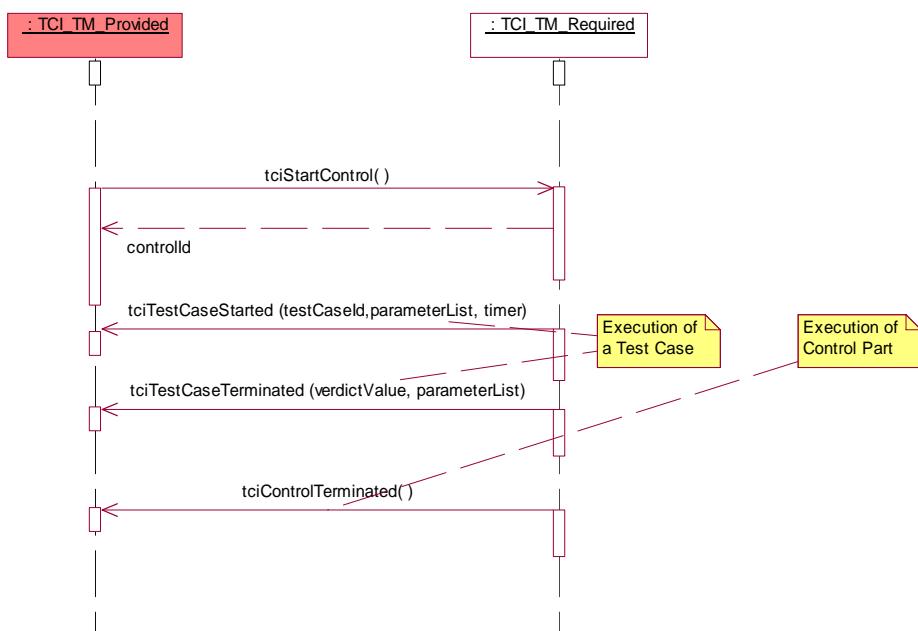
Figure 13 – Use scenario – test case execution within control

11.2.2.2   TTCN-3 fragment

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.2.3   Use scenario: direct test case execution

The scenario in Figure 14 shows how a test case can be directly executed from the test management outside the control part. After selecting the TTCN-3 module containing the test case to be executed, the start of the test case is requested. When the test case completes its execution, the test management is informed by the TE of the test case termination.

11.2.3.1 Sequence diagram
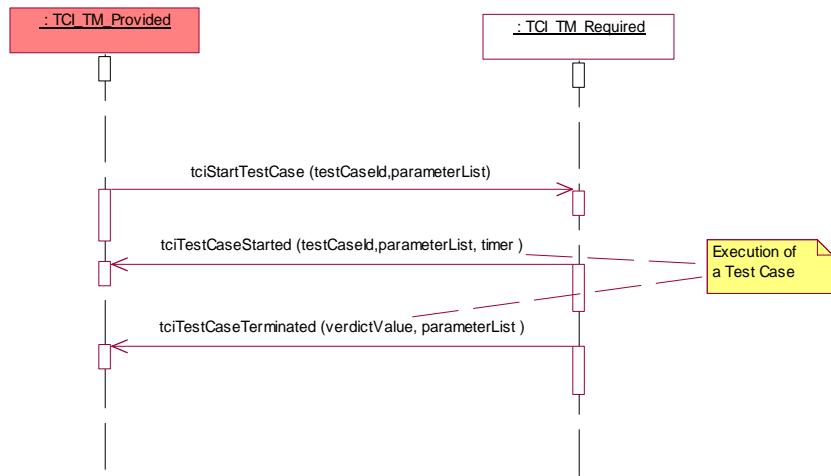
Figure 14 – Use scenario – direct test case execution

11.2.3.2 TTCN-3 fragment

The direct execution of a test case is outside the scope of TTCN-3.

11.2.4 Use scenario: execute test case to TRI

The scenario in Figure 15 shows how the TRI is informed about the execution of a test case so that it can set up and initialize system ports when needed. The execute test case request has to be issued before the test behaviour on the MTC of the current test case is started.

11.2.4.1 Sequence diagram
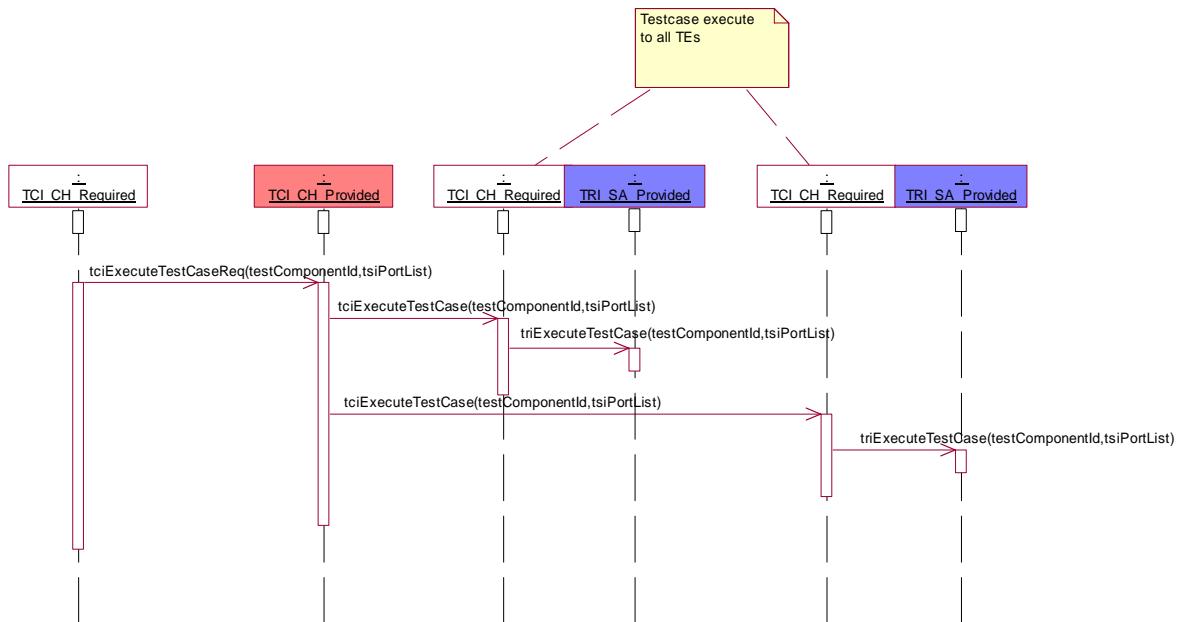
Figure 15 – Use scenario – execute test case to TRI

11.2.4.2 TTCN-3 fragment

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3 Component handling

11.3.1 Use scenario: local control component creation

The scenario in Figure 16 demonstrates the creation of the control component on the same node where the user interface to the test management TCI-TM resides. A control component is created whenever the control part of a TTCN-3 module is executed. Whenever the test management TCI-TM issues the start of the control part, a create test component request is sent to the TCI-CH, which propagates it to the TE where the control component should be created. In this case it is the TE on the same node. The identifier for the control component is returned and given to the TCI-TM. The identifier is then used to start the behaviour of the control part on the control component.

11.3.1.1 Sequence diagram
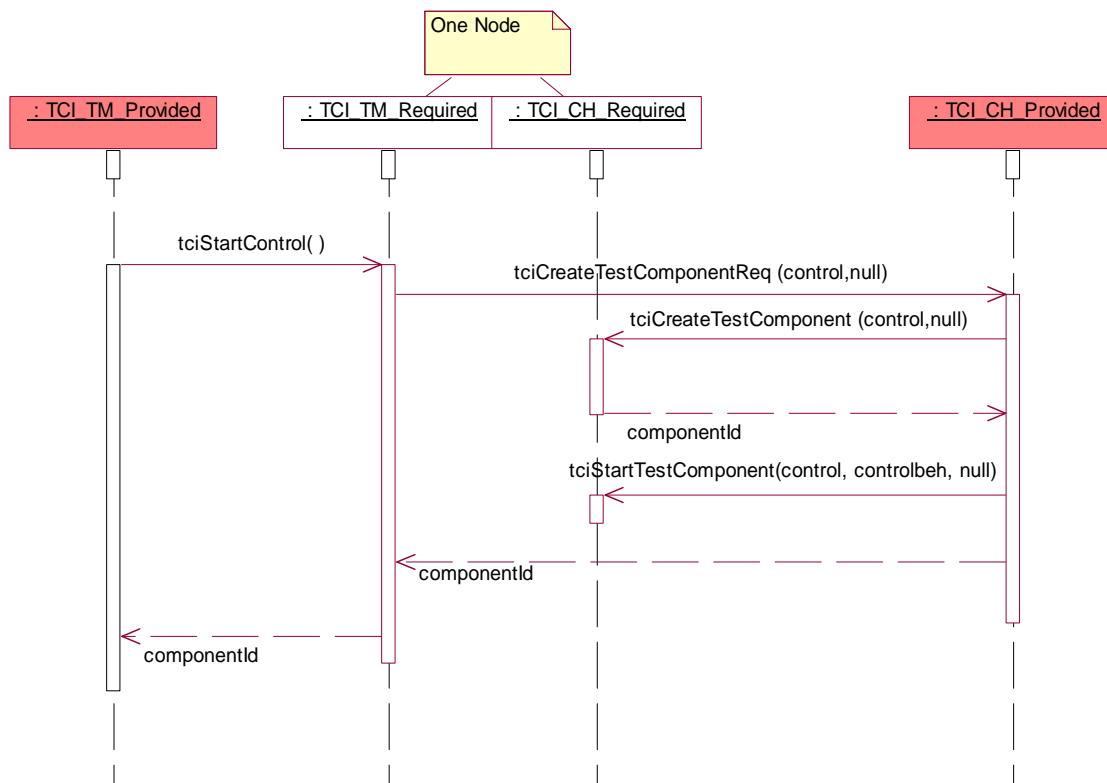
Figure 16 – Use scenario – local control component creation

11.3.1.2 TTCN-3 fragment

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.3.2 Use scenario: remote control component creation

The scenario in Figure 17 demonstrates the creation of the control component on another node than where the user interface to the test management TCI-TM resides. A control component is created whenever the control part of a TTCN-3 module is executed. Whenever the test management TCI-TM issues the start of the control part, a create test component request is sent to the TCI-CH, which propagates it to the TE where the control component should be created. In this case it is the TE on another remote node. The identifier for the control component is returned and given to the TCI-TM. The identifier is then used to start the behaviour of the control part on the control component.

11.3.2.1 Sequence diagram
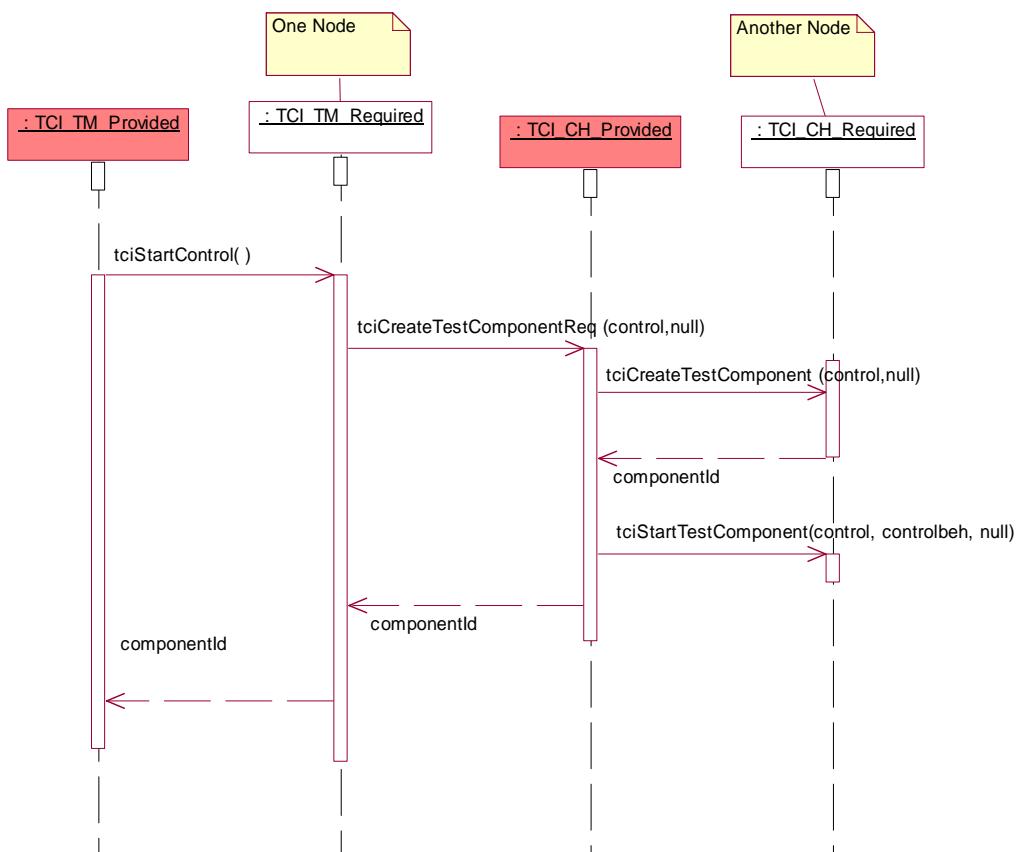
Figure 17 – Use scenario – remote control component creation

11.3.2.2 TTCN-3 fragment

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.3.3 Use scenario: local MTC creation

The scenario in Figure 18 demonstrates the local creation of the main test component. Local is meant for two cases:

1)   on the same node where the user interface to the test management TCI-TM resides (when a test case is started directly); or

2)   on the same node where the control component resides (when a test case is executed from a control part).

A main test component is created whenever a test case is executed: a create test component request is sent to the TCI-CH, which propagates it to the TE where the main test component should be created. In this case it is the TE on the same node. The identifier for the main test component is returned and given to the TCI-TM. The identifier is then used to start the test case behaviour on the main test component (this is not shown here, but handled the same way as in the scenarios described in clauses 11.3.5 and 11.3.6).

11.3.3.1 Sequence diagram
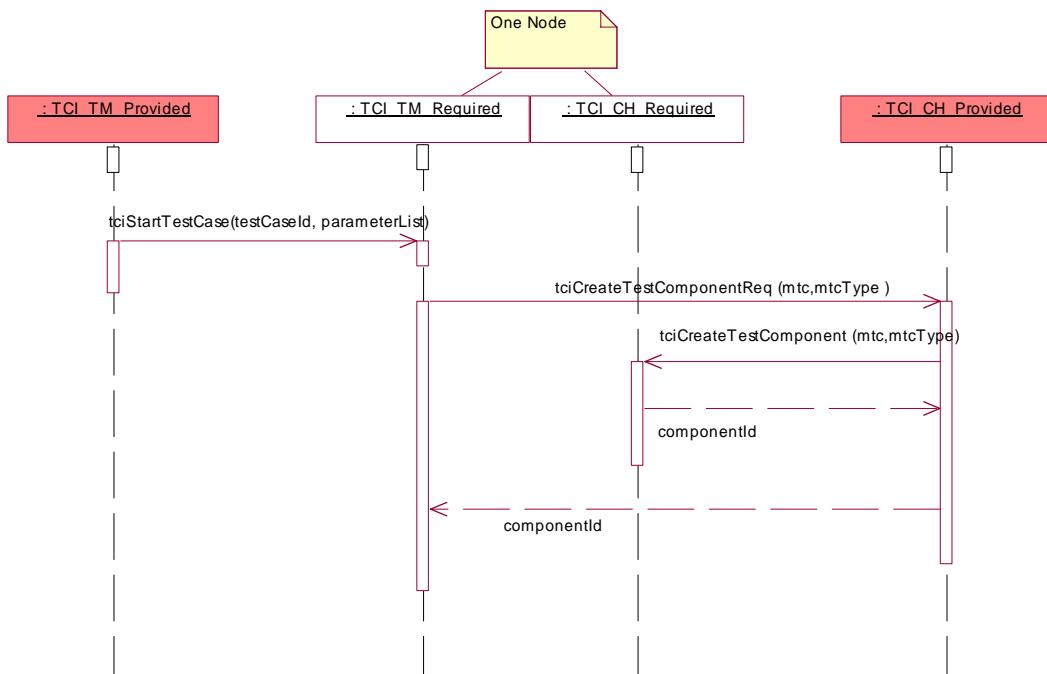
Figure 18 – Use scenario – local MTC creation

11.3.3.2 TTCN-3 fragment

```
module AModule {
    …
    testcase ATestCase (…)runs on MTCType… {
        … //the test case behaviour
    }
    …
}
```

11.3.4 Use scenario: remote MTC creation

The scenario in Figure 19 demonstrates the remote creation of the main test component. Remote is meant for two cases:

1)   on another node than where the user interface to the test management TCI-TM resides (when a test case is started directly); or

2)   on another node than where the control component resides (when a test case is executed from a control part).

A main test component is created whenever a test case is executed: a create test component request is sent to the TCI-CH, which propagates it to the TE where the main test component should be created. In this case it is the TE on another node. The identifier for the main test component is returned and given to the TCI-TM. The identifier is then used to start the test case behaviour on the main test component (this is not shown here, but handled the same way as in the scenarios described in clauses 11.3.5 and 11.3.6).

11.3.4.1 Sequence diagram
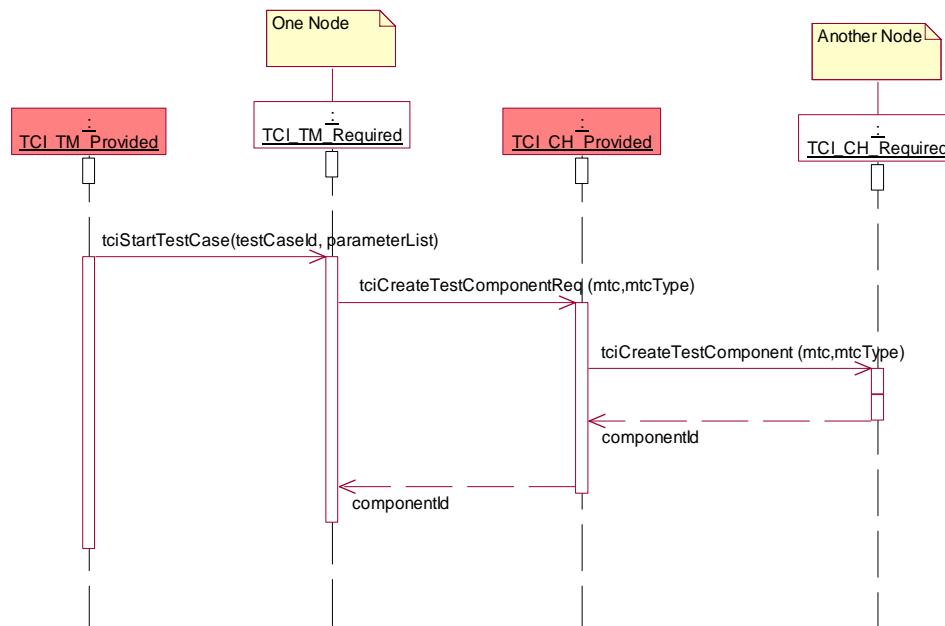
Figure 19 – Use scenario – remote MTC creation

11.3.4.2 TTCN-3 fragment

```
module AModule {
    …
    testcase ATestCase(…) runs on MTCType … {
        … //the test case behaviour
    }
    …
}
```

11.3.5 Use scenario: component handling for test case execution within control

The scenario in Figure 20 demonstrates the handling of components for the test case execution within a control part. When the control part is started, a control component is created and its component identifier returned to the test management. For each test case to be executed within the control part, a main test component is created and the component identifier returned to the control component. Afterwards, the test case behaviour is started on the main test component and the test management is informed about the start of the test case. When the main test component terminates, a request for the main test component termination together with the local verdict of the main test component is propagated to enable the derivation of the global test verdict and to enable the information about the test case termination.

11.3.5.1 Sequence diagram
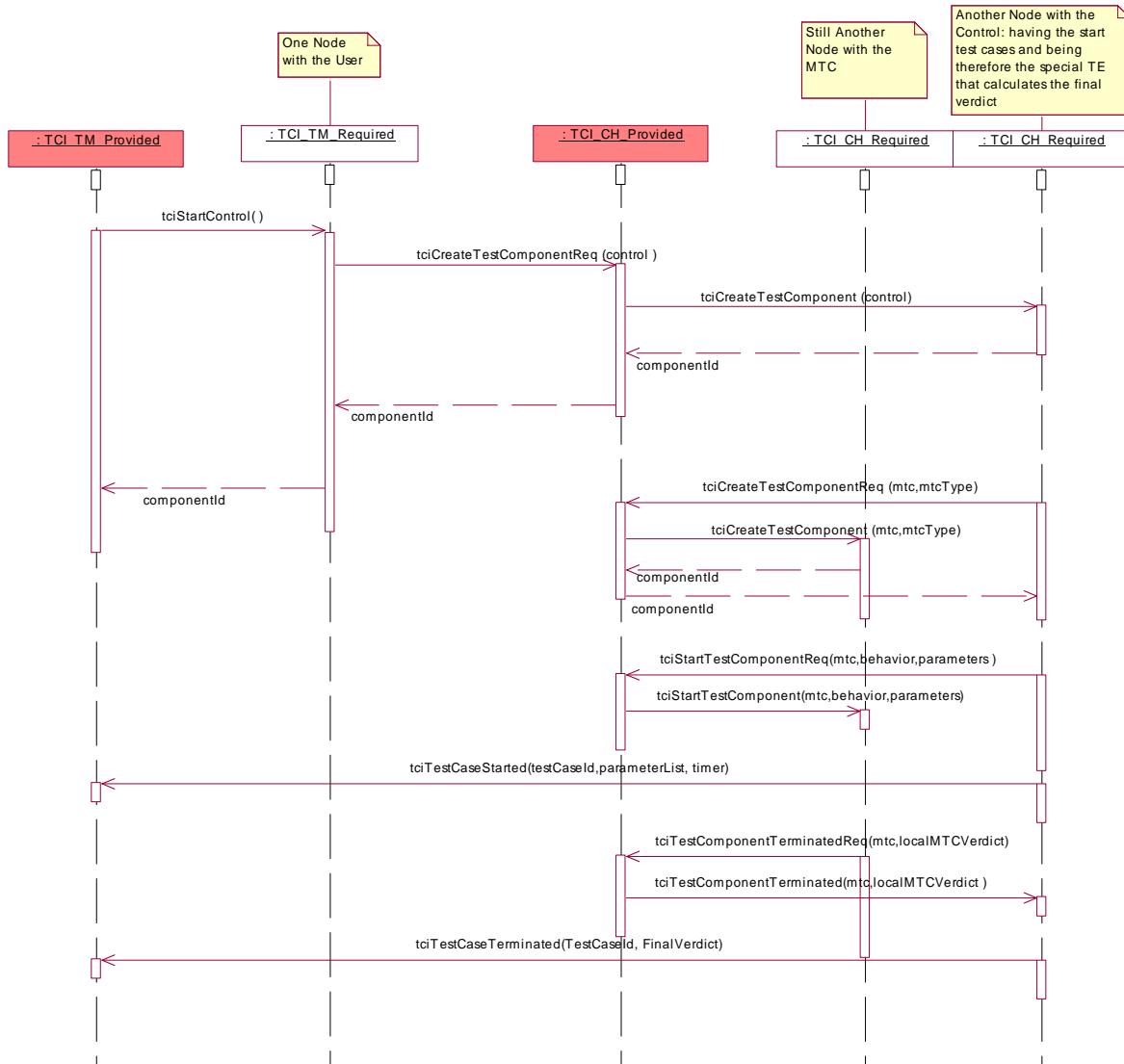
Figure 20 – Use scenario – Component handling for test case execution within control

11.3.5.2 TTCN-3 fragment

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3.6 Use scenario: component handling for direct test case execution

The scenario in Figure 21 shows how test components are handled when a test case is executed directly, i.e., outside a control part. When a test case is started, the main test component is created and the test case behaviour started on this main test component at first. Whenever a parallel test component is used within a test case, it is handled the same: the parallel test component is started first: giving a test component create request to the TCI-CH entity, which propagates the test component create to the TE in which the parallel test component shall be created. The identifier for the created parallel test component is returned. The identifier is then used to start the PTC behaviour for the start operation. When the PTC terminates its execution, a test component terminate request together with the local test verdict is issued to inform TCI-CH about this termination. The same is done when the main test component terminates. In addition, the termination of the main test component leads to the overall termination of the test case.

11.3.6.1 Sequence diagram
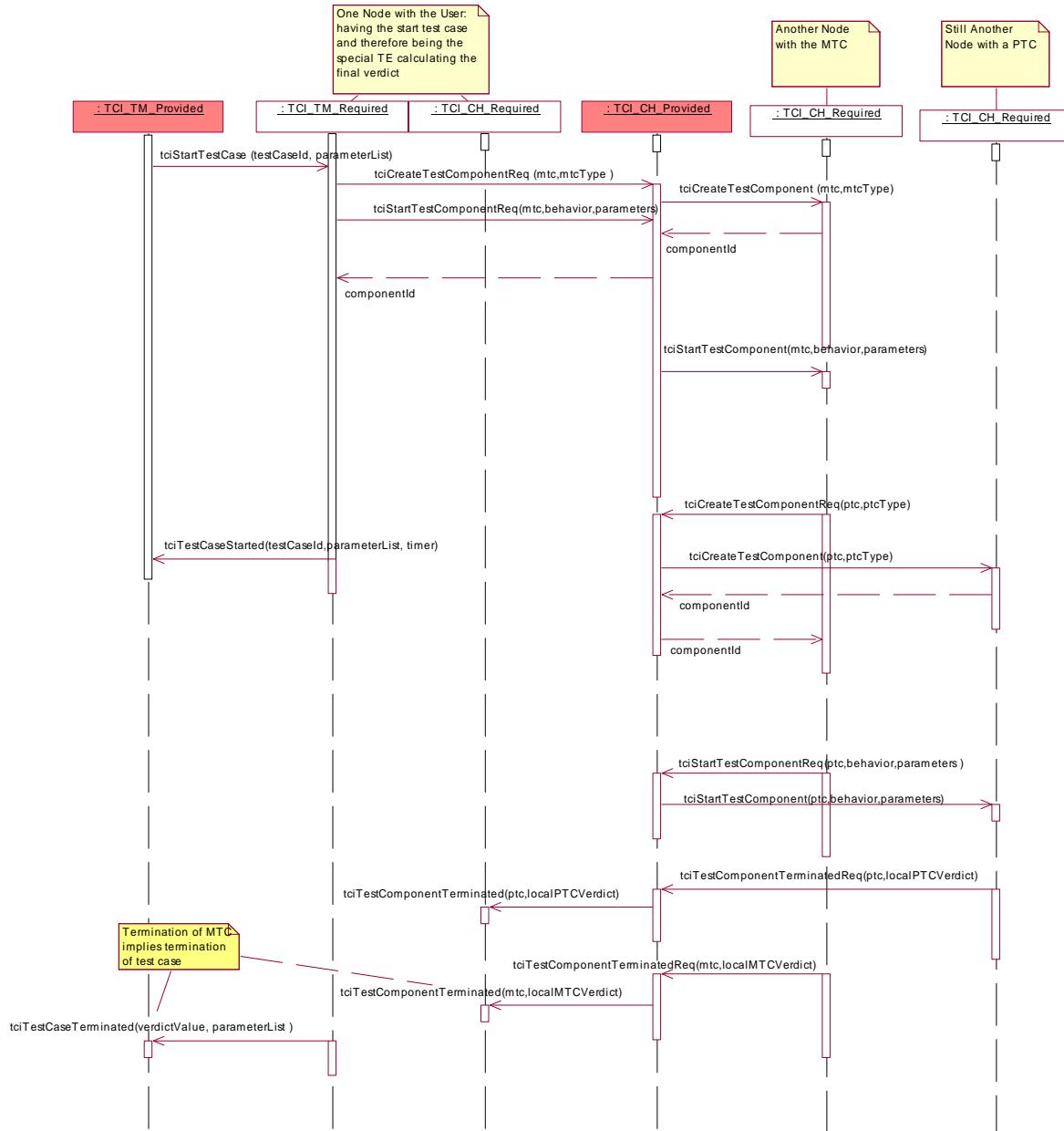
Figure 21 – Use scenario – Component handling for direct test case execution

11.3.6.2   TTCN-3 fragment

```
module AModule {
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    }
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
        var APTCType PTC:= APTCType.create;
        …
        PTC.start(APTCBehaviour(…));
        …
    }
    …
}
```

11.3.7   Use scenario: propagation of map/connect

The scenario in Figure 22 shows how ports are mapped. The request to map a port is propagated to the TE where the map is finally performed. The propagation of connect requests works analogously.

11.3.7.1   Sequence diagram
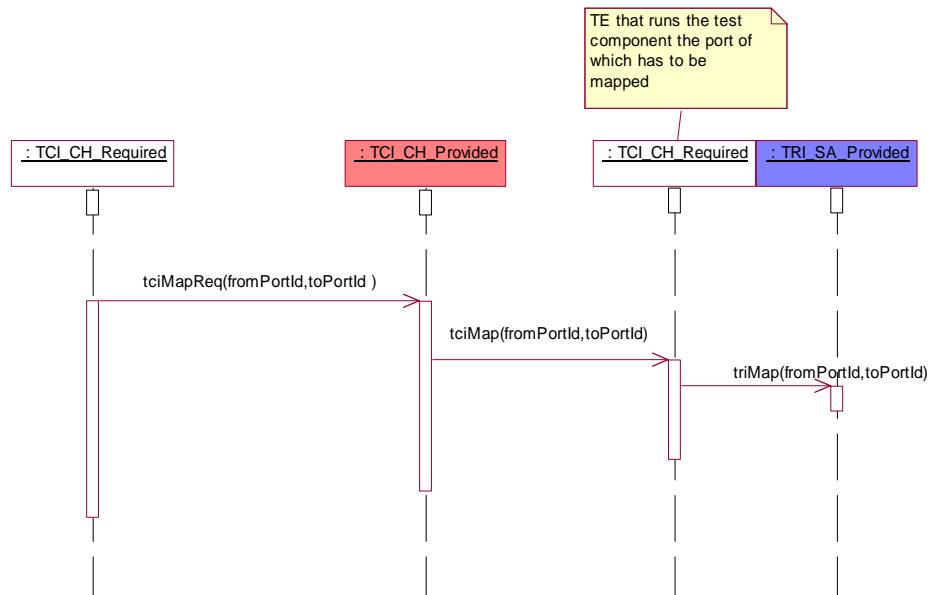
Figure 22 – Use scenario – propagation of map

11.3.7.2   TTCN-3 fragment

```
module AModule {
    …
    type port A { … }
    type component CA { port A a }
        type component CB { port A a }
    …
    testcase ATestCase(…) runs on CA system CB {
        var CA ptc := CA.create;
        … //the test case behaviour
        map(ptc:a,system:a);
        …
    }
    …
}
```

11.3.8 Use scenario: propagation of unmap/disconnect

The scenario in Figure 23 shows how ports are unmapped. The request to unmap a port is propagated to the TE where the unmap is finally performed. The propagation of disconnect requests works analogously.

11.3.8.1 Sequence diagram
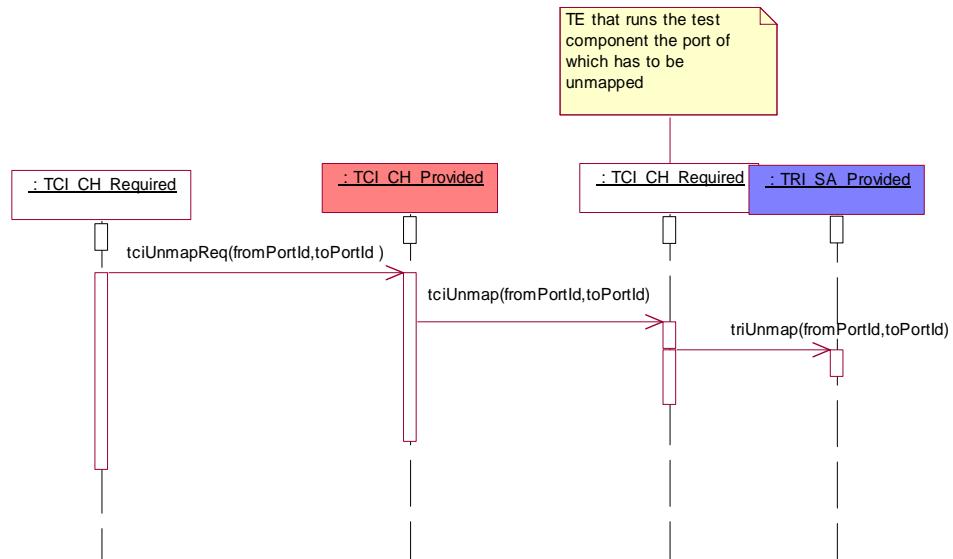
Figure 23 – Use scenario – propagation of unmap

11.3.8.2 TTCN-3 fragment

```
module AModule {
    …
    type port A { … }
    type component CA { port A a }
    type component CB { port A a }
    …
    testcase ATestCase(…)runs on CA system CB {
        var CA ptc := CA.create;
        … //the test case behaviour
        unmap(ptc:a,system:a);
        …
    }
    …
}
```

11.4 Termination of test cases and control

11.4.1 Use scenario: stop a test case

The scenario in Figure 24 shows how a test case is stopped from the test management during test case execution. Once the TM has received information about a started test case, a stop test case can be requested up until receiving the information that the test case has been terminated. Upon stopping a test case, all parallel test components will be stopped and the test system will be reset.

11.4.1.1 Sequence diagram
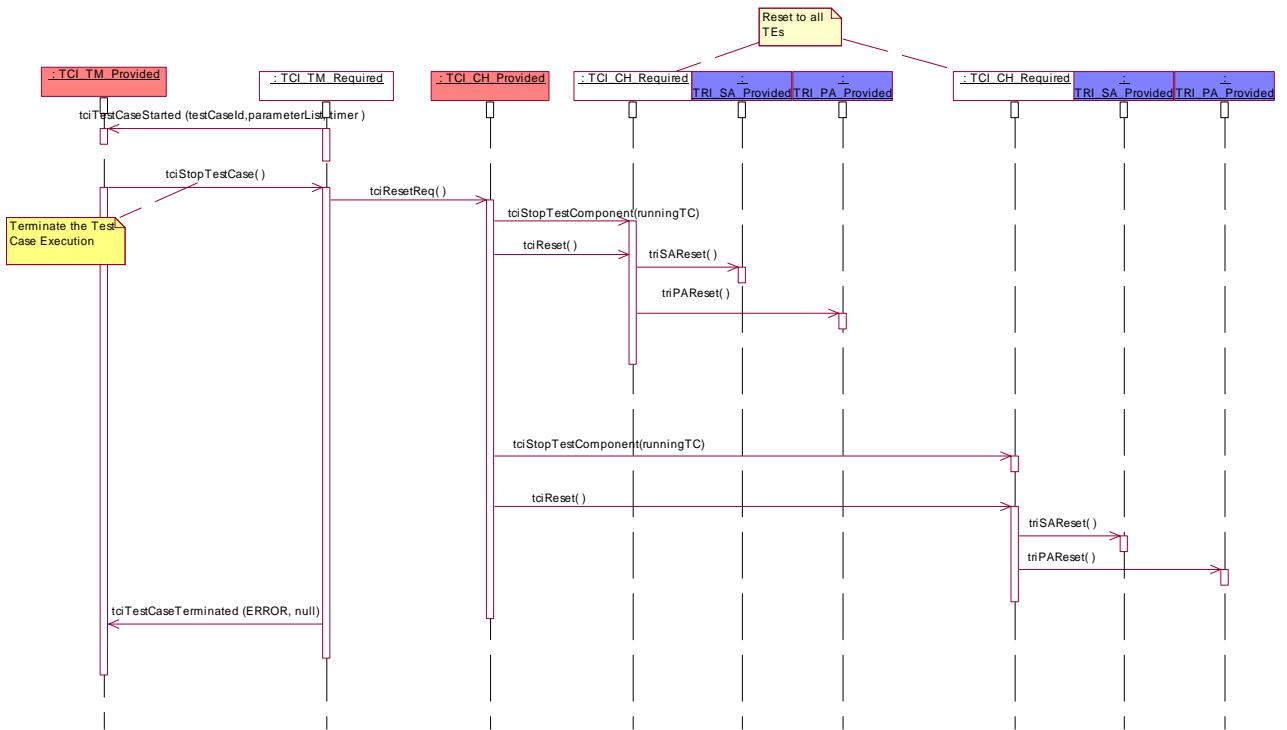
Figure 24 – Use scenario – stop a test case

11.4.1.2 TTCN-3 fragment

There is no TTCN-3 code related to how the TM chooses to implement test case termination. This is outside the scope of TTCN-3.

11.4.2 Use scenario: stop control

The scenario in Figure 25 shows how a control part is stopped from the test management during control part execution. A control part can be stopped in between starting the control and its termination. If the control part receives a stop test case request while a test case is executing, the executing test case shall be stopped. Furthermore, the test system shall be reset as described in Figure 24.

11.4.2.1 Sequence diagram
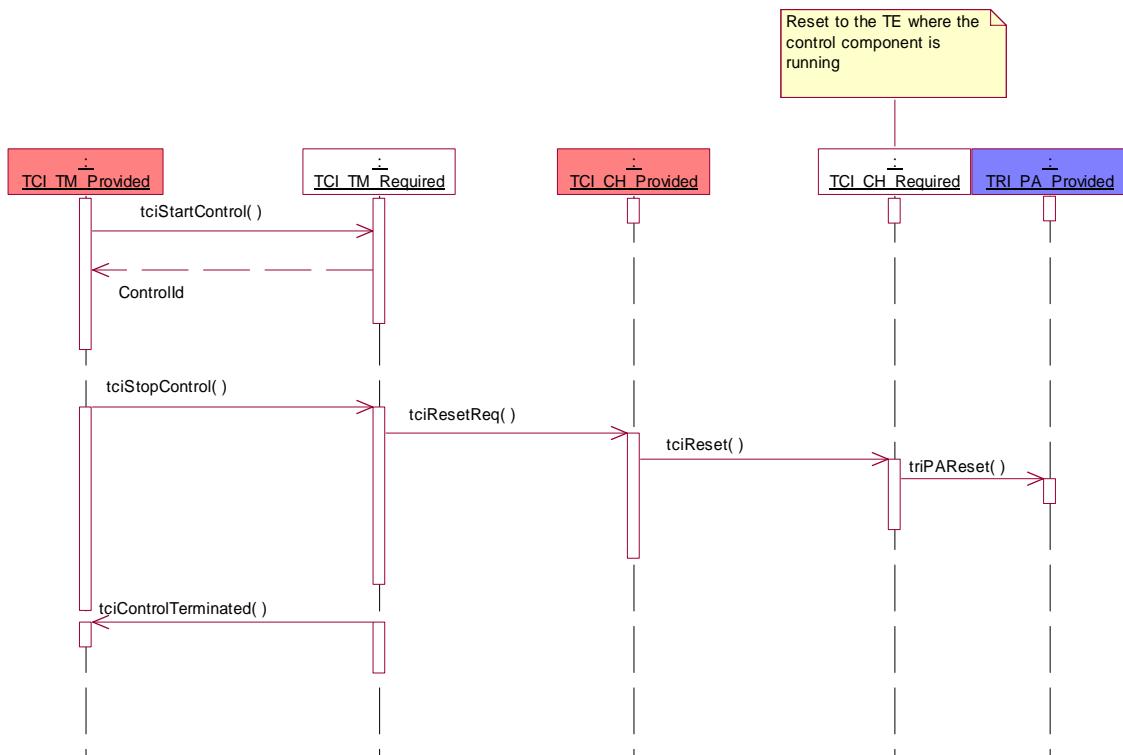
Figure 25 – Use scenario – stop control

11.4.2.2 TTCN-3 fragment

Stopping a control part from the test management is outside the scope of TTCN-3 so that no TTCN-3 fragment exists.

11.4.3 Use scenario: termination of control after error

The scenario in Figure 26 shows the handling of error situations during the execution of a control part when no test case is being executed. The test management is informed about the error situation and has then to terminate the execution of the control part explicitly. Upon termination of the control part, the test system will be reset.

11.4.3.1 Sequence diagram
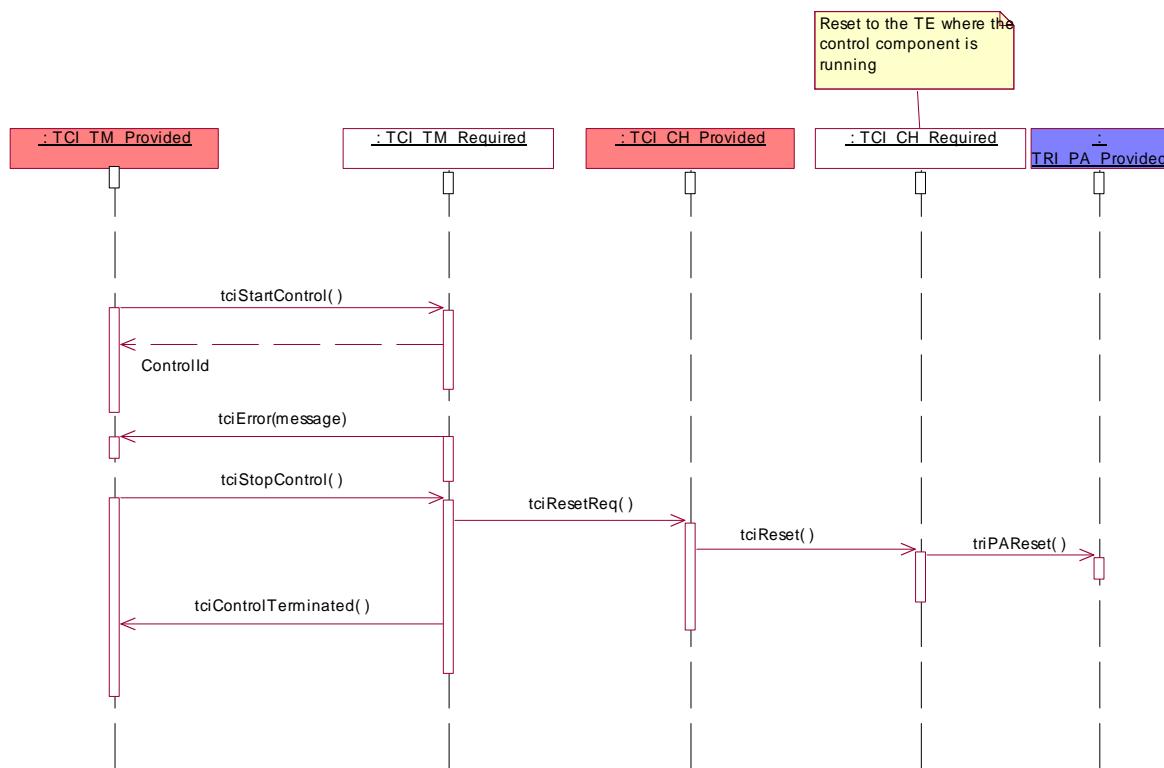
Figure 26 – Use scenario – termination of control after error

11.4.3.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since error situations are exceptional cases in a test system and not a TTCN-3 concept as such. Rather, the TTCN-3 semantics describes various potential error situations in a test system.

11.4.4 Use scenario: termination of a test case after error

The scenario in Figure 27 shows the handling of error situations during the direct execution of a test case. The test management is informed about the error situation. The TM must then explicitly terminate test case execution. Upon stopping a test case, the parallel test components will be stopped and the test system shall be reset.

11.4.4.1   Sequence diagram
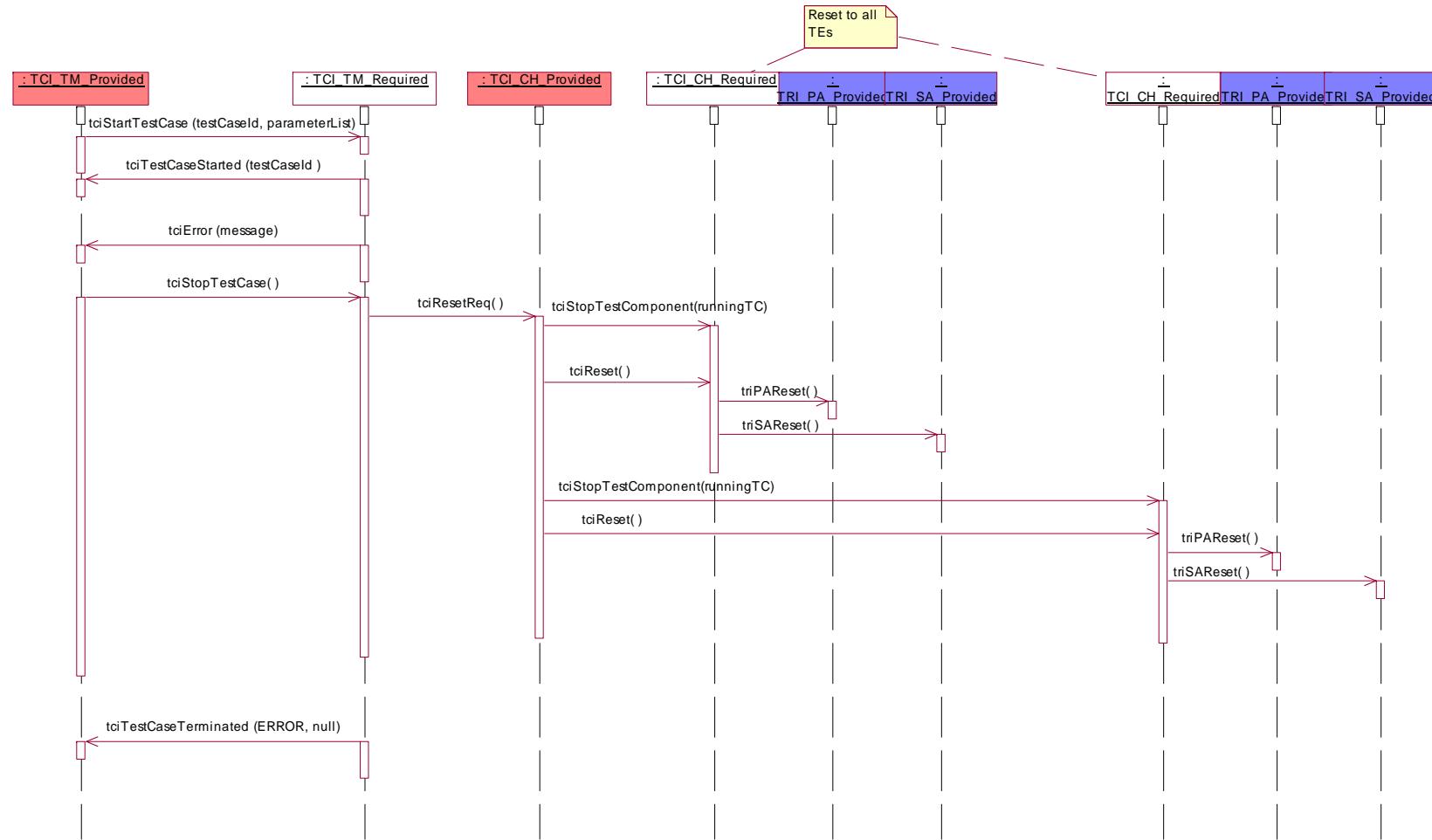
Figure 27 – Use scenario – termination of a test case after error

11.4.4.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since error situations are exceptional cases in a test system and not a TTCN-3 concept as such. Rather, the TTCN-3 semantics describes various potential error situations in a test system.

11.4.5 Use scenario: reset

The scenario in Figure 28 shows the reset of the test system. In that case, all involved TEs together with their TRI System Adaptors (SA) and Platform Adaptors (PA) are reset.

11.4.5.1 Sequence diagram
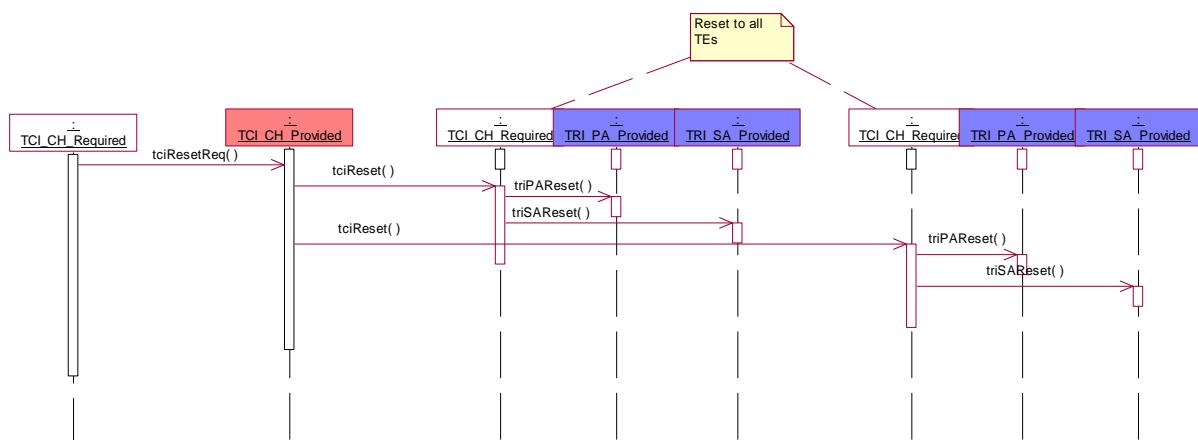
Figure 28 – Use scenario – reset

11.4.5.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since reset as required after error situations are exceptional cases in a test system and not a TTCN-3 concept as such.

11.5 Communication

11.5.1 Use scenario: local intercomponent communication

The scenario in Figure 29 shows the communication between test components (main test component or parallel test components), which reside on the same node. A communication request is given to the TCI-CH, which then decide where to enqueue this communication template. In this case, the communication is done locally via the TE on the same node. The scenario shows a message-based communication using the send operation – the scenario is the same for call, reply, and raise operations.

11.5.1.1 Sequence diagram
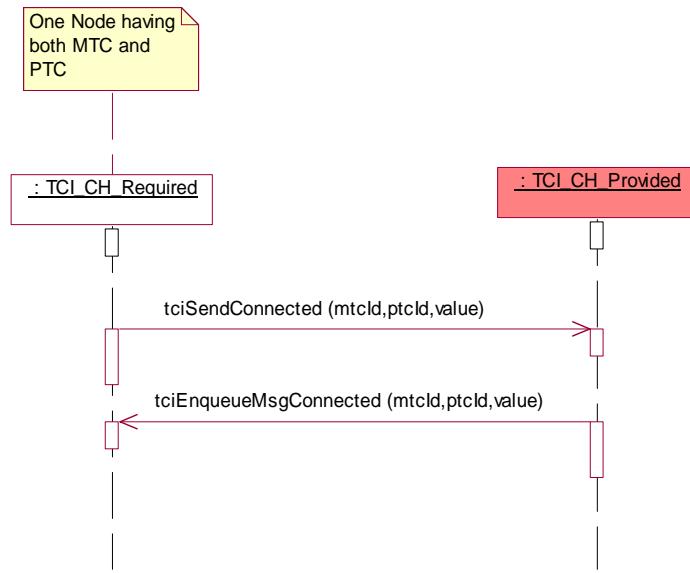
Figure 29 – Use scenario – local intercomponent communication

11.5.1.2 TTCN-3 fragment

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //sending data to a test component
        …
    }
    …
}
```

11.5.2 Use scenario: internode communication between test components

The scenario in Figure 30 shows the communication between test components (main test component or parallel test components), which reside on different nodes. A communication request is given to the TCI-CH, which then decides where to enqueue this communication template. In this case, the communication is done remotely via the TE on another node. The scenario shows a message-based communication using the send operation – the scenario is the same for call, reply, and raise operations.

11.5.2.1   Sequence diagram
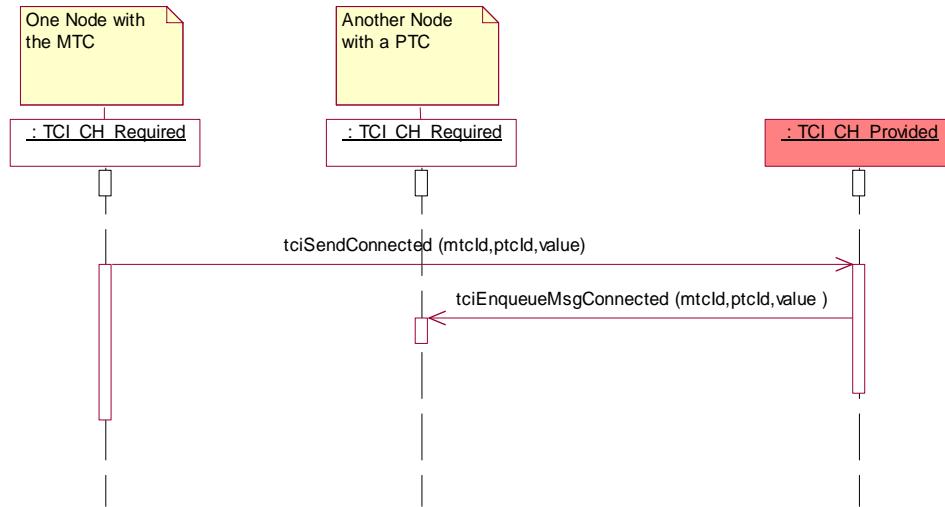
Figure 30 – Use scenario – internode communication between test components

11.5.2.2   TTCN-3 fragment

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //sending data to a test component
        …
    }
    …
}
```

11.5.3   Use scenario: encoding

The scenario in Figure 31 shows the encoding of data, which is sent to the SUT. The encoded data is received from the coding/decoding entity via the TCI-CD. The encoded value is sent to the SUT via the TRI-SA. The scenario is the same for the call, the reply, and the raise operations.

11.5.3.1 Sequence diagram
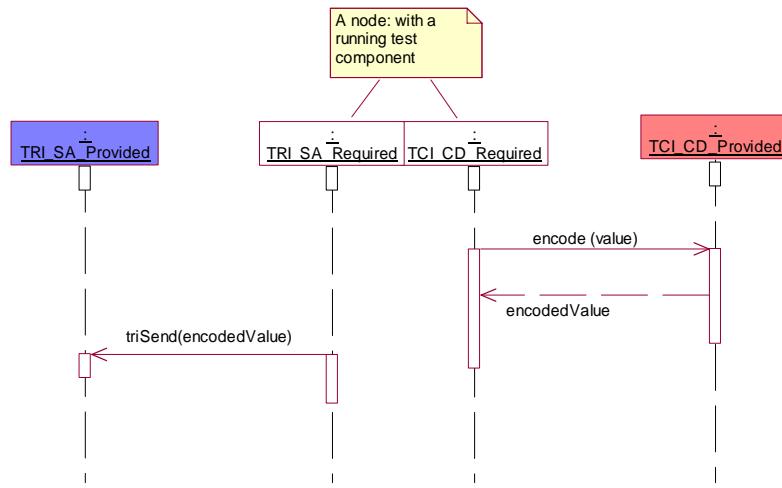
Figure 31 – Use scenario – encoding

11.5.3.2 TTCN-3 fragment

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //the test case behaviour
        map(mtc:APort,system:APort);
        …
        APort.send(AMessageTemplate); //sending data to the SUT
        …
    }
    …
} with { encoding = '…' }
```

11.5.4 Use scenario: decoding

The scenario in Figure 32 shows the decoding of data, which is received from the SUT via the TRI-SA. The decoded data is received from the coding/decoding entity via the TCI-CD. The scenario is the same for the receive, the getcall, the getreply, the catch, and the check operations.

11.5.4.1 Sequence diagram
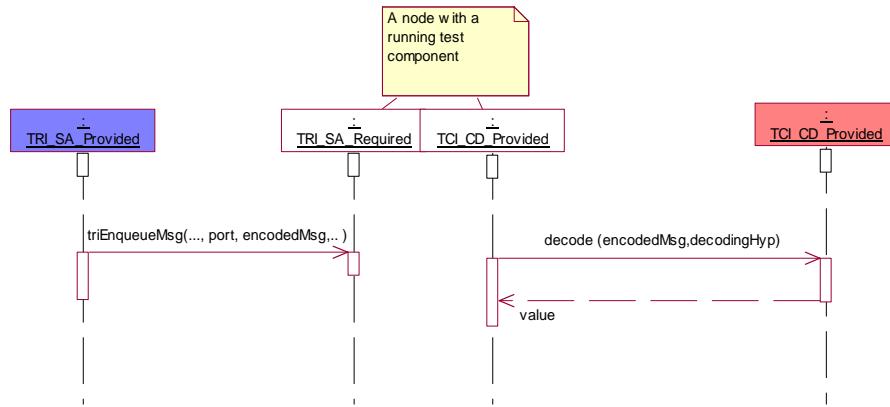
Figure 32 – Use scenario – decoding

11.5.4.2 TTCN-3 fragment

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //the test case behaviour
        map(mtc:APort,system:APort);
        …
        APort.receive(AMessageTemplate); //receiving data from the SUT
        …
    }
    …
} with { encoding = '…' }
```

Annex A

IDL specification of TCI

(This annex forms an integral part of this Recommendation)

This annex defines the TTCN-3 Control Interfaces using the Interface Definition Language (IDL).

```
// ****************************************************************************
// * Interface definitions for the TTCN-3 Control Interfaces
// ****************************************************************************

module tciInterface {

  /* Forward declaration */
  interface Value;
  interface Type;

  // ****************************************************************************
  // * Data types taken from the TRI definitions
  // ****************************************************************************

  // Connection
  native TriPortIdType ;
  native TriPortIdListType;
  native TriComponentIdType ;
native TriComponentIdListType;

  // Communications
  native TriMessageType;
  native TriParameterType;
  native TriParameterListType;
  native TriAddressType;
  native TriAddressListType;
  native TriExceptionType;
  native TriSignatureIdType;

  // Miscellaneous
  native TriStatusType;
  native TriTimerIdType;
  native TriTimerDurationType;

  native TciStatusType;

  // ****************************************************************************
  // * General Abstract Data Types
  // ****************************************************************************

  // Basic definitions
  native TBoolean;
  native TFloat;
  native TChar;
  native TInteger;
  native TString;
  native TUniversalChar;
  typedef sequence <TString> TStringSeq;
  native TObjid;

  struct QualifiedName {
    TString moduleName;
    TString baseName;
  };

  // General TCI abstract data types
  typedef QualifiedName TciBehaviourIdType;
  typedef QualifiedName TciModuleIdType;
  typedef QualifiedName TciModuleParameterIdType;
  typedef QualifiedName TciTestCaseIdType;

  enum TciParameterPassingModeType {
    IN_MODE,
    OUT_MODE,
    INOUT_MODE
  };
```

```
struct TciParameterType {
  TciModuleParameterIdType parameterName;
  Value parameterValue;
  TciParameterPassingModeType mode;
};

typedef sequence <TciParameterType> TciParameterListType;

struct TciParameterTypeType {
  Type parameterType;
  TciParameterPassingModeType mode;
};

typedef sequence <TciParameterTypeType> TciParameterTypeListType;

struct TciModuleParameterType {
  TciModuleParameterIdType parameterName;
  Value defaultValue;
};

typedef sequence <TciModuleIdType> TciModuleIdListType ;

typedef sequence <TciModuleParameterType> TciModuleParameterListType;

typedef sequence <TciTestCaseIdType> TciTestCaseIdListType;


enum TciTestComponentKindType {
  CONTROL,
  MTC,
  PTC,
  SYSTEM,
  PTC_ALIVE
};

enum ComponentStatusType{
  inactiveC,
  runningC,
  stoppedC,
  killedC
};

enum TimerStatusType{
  runningT,
  inactiveT,
  expiredT
};

enum PortStatusType{
  startedP,
  haltedP,
  stoppedP
};

enum TciTypeClassType {
  ADDRESS_CLASS,
  ANYTYPE_CLASS,
  BITSTRING_CLASS,
  BOOLEAN_CLASS,
  CHAR_CLASS,
  CHARSTRING_CLASS,
  COMPONENT_CLASS,
  ENUMERATED_CLASS,
  FLOAT_CLASS,
  HEXSTRING_CLASS,
  INTEGER_CLASS,
  OBJID_CLASS,
  OCTETSTRING_CLASS,
  RECORD_CLASS,
  RECORDOF_CLASS,
  SET_CLASS,
  SETOF_CLASS,
  UNION_CLASS,
  UNIVERSALCHAR_CLASS,
  UNIVERSALCHARSTRING_CLASS,
  VERDICT_CLASS
};
```

```
// **************************************************************************
// * Abstract TTCN-3 Data Types And Values
// **************************************************************************

// Abstract data type "Type"
interface Type {
  TciModuleIdType  getDefiningModule ();
  TString          getName ();
  TciTypeClassType getTypeClass ();
  Value            newInstance ();
  TString          getTypeEncoding ();
  TString          getTypeEncodingVariant ();
  TStringSeq       getTypextension ();
};

// Abstract TTCN-3 Values
interface Value {
  TString  getValueEncoding ();
  TString  getValueEncodingVariant ();
  Type     getType ();
  TBoolean  notPresent ();
};

interface RecordOfValue : Value {
  Value    getField (in TInteger position);
  void     setField (
           in TInteger position,
           in Value value
             );
  void     appendField (in Value value);
  Type     getElementType ();
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface RecordValue : Value {
  Value     getField (in TString fieldName);
  void      setField (
           in TString fieldName,
           in Value value
           );
  TStringSeq getFieldNames ();
  void       setFieldOmitted (in TString fieldName);
};

interface VerdictValue : Value {
  TInteger getVerdict ();
  void     setVerdict (in TInteger verdict);
};

interface BitstringValue : Value {
  TString   getString ();
  void      setString (in TString value);
  TInteger  getBit (in TInteger position);
  void      setBit (
           in TInteger position,
           in TInteger value
           );
  TInteger  getLength ();
  void      setLength (in TInteger len);
};

interface OctetstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TInteger getOctet (in TInteger position);
  void     setOctet (
           in TInteger position,
           in TInteger value
           );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface FloatValue : Value {
  TFloat getFloat ();
  void  setFloat (in TFloat value);
};
```

```
interface HexstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TInteger getHex (in TInteger position);
  void     setHex (
           in TInteger position,
           in TInteger value
           );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface ObjidValue : Value {
  TObjid getObjid ();
  void   setObjid (in TObjid value);
};

interface EnumeratedValue : Value {
  void    setEnum (in TString enumValue);
  TString getEnum ();
};

interface IntegerValue : Value {
  TInteger getInt ();
  void     setInt (in TInteger value);
};

interface CharValue : Value {
  TChar getChar ();
  void  setChar (in TChar value);
};

interface CharstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TChar    getChar   (in TInteger position);
  void     setChar   (
           in TInteger position,
           in TChar value
           );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface BooleanValue : Value {
  TBoolean getBoolean ();
  void     setBoolean (in TBoolean value);
};

interface UniversalCharValue : Value {
  TUniversalChar getUniversalChar ();
  void           setUniversalChar (in TUniversalChar value);

};

interface UniversalCharstringValue : Value {
  TString        getString ();
  void           setString (in TString value);
  TUniversalChar getChar   (in TInteger position);
  void           setChar   (
           in TInteger position,
           in TUniversalChar value
           );
  TInteger       getLength ();
  void           setLength (in TInteger len);
};

interface UnionValue : Value {
  Value     getVariant  (in TString variantName);
  void      setVariant  (
           in TString variantName,
           in Value value
           );
  TString   getPresentVariantName ();
  TStringSeq getVariantNames ();

};
// ************************************************************************
// * Abstract Logging Types
// ************************************************************************
```

```
interface TciValueTemplate : Value {
    boolean isOmit ();
    boolean isAny();
    boolean isAnyOrOmit();
    TString getTemplateDef();
};

interface TciNonValueTemplate {
    boolean isAny();
    boolean isAll();
    TString getTemplateDef();
};

typedef sequence <Value> TciValueList;

struct TciValueDifference
{
  TString desc;
  Value val;
  TciValueTemplate tmpl;
};

typedef sequence <TciValueDifference> TciValueDifferenceList;


// ********************************************************************************
// Coding Decoding Interface
// -  Required
// ********************************************************************************

interface TCI_CD_Required {
  Type getTypeForName (in TString typeName);
  Type getInteger ();
  Type getFloat ();
  Type getBoolean ();
  Type getChar ();
  Type getUniversalChar ();
  Type getObjid ();
  Type getCharstring ();
  Type getUniversalCharstring ();
  Type getHexstring ();
  Type getBitstring ();
  Type getOctetstring ();
  Type getVerdict ();
  void tciErrorReq (in TString message);
};

// ********************************************************************************
// Coding Decoding interface
// -  Provided
// ********************************************************************************

interface TCI_CD_Provided {
  Value decode (
        in TriMessageType message,
        in Type decodingHypothesis
        );
  TriMessageType encode (in Value value);
};


// ********************************************************************************
// Test Management Interface
// -  Required
// ********************************************************************************

interface TCI_TM_Required : TCI_CD_Required {
  void tciRootModule (in TciModuleIdType moduleName);
  TciModuleIdListType getImportedModules();
  TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName);
  TciTestCaseIdListType tciGetTestCases ();
  TciParameterTypeListType tciGetTestCaseParameters (
        in TciTestCaseIdType testCaseId
        );
  TriPortIdListType tciGetTestCaseTSI (
        in TciTestCaseIdType testCaseId
        );
  void tciStartTestCase (
        in TciTestCaseIdType testCaseId,
```

```
            in TciParameterListType parameterList
            );
    void tciStopTestCase ();
    TriComponentIdType tciStartControl ();
    void tciStopControl ();
};

// ******************************************************************************
// Test Management Interface
// -  Provided
// ******************************************************************************

interface TCI_TM_Provided {
    void tciTestCaseStarted (
            in TciTestCaseIdType testCaseId,
            in TciParameterListType parameterList,
            in TFloat timer
            );
    void tciTestCaseTerminated (
            in VerdictValue verdict,
            in TciParameterListType parameterList
            );
    void tciControlTerminated ();
    Value tciGetModulePar (
            in TciModuleParameterIdType parameterId
            );
    void tciLog (
        in TriComponentIdType testComponentId,
        in TString message
        );
    void tciError (in TString message);
};


// ******************************************************************************
// Component Handling Interface
// -  Required
// ******************************************************************************

interface TCI_CH_Required : TCI_CD_Required {
    void tciEnqueueMsgConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in Value receivedMessage
            );
    void tciEnqueueCallConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList
            );
    void tciEnqueueReplyConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList,
            in Value returnValue
            );
    void tciEnqueueRaiseConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in TriSignatureIdType signature,
            in Value except
            );
    TriComponentIdType tciCreateTestComponent (
            in TciTestComponentKindType kind,
            in Type componentType,
            in TString name
            );
    void tciStartTestComponent (
            in TriComponentIdType comp,
            in TciBehaviourIdType behavior,
            in TciParameterListType parameterList
            );
    void tciStopTestComponent (
            in TriComponentIdType comp
            );
    void tciConnect (
            in TriPortIdType fromPort,
            in TriPortIdType toPort
```

```
            );
     void tciDisconnect (
            in TriPortIdType fromPort,
            in TriPortIdType toPort
            );
     void tciTestComponentTerminated (
            in TriComponentIdType comp,
            in VerdictValue verdict
            );
     TBoolean tciTestComponentRunning (
            in TriComponentIdType comp
            );
     TriComponentIdType tciGetMTC ();
     void tciMap (
            in TriPortIdType fromPort,
            in TriPortIdType toPort
            );
     void tciUnmap (
            in TriPortIdType fromPort,
            in TriPortIdType toPort
            );
     void tciExecuteTestCase (
            in TciTestCaseIdType testCaseId,
            in TriPortIdListType tsiPortList
            );
     TBoolean tciTestComponentDone (
            in TriComponentIdType comp
            );
     void tciReset ();
};

// ******************************************************************************
// Component Handling Interface
// -   Provided
// ******************************************************************************

interface TCI_CH_Provided {
  void tciSendConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in Value sendMessage
            );
  void tciSendConnectedBC (
            in TriPortIdType sender,
            in Value sendMessage
            );
  void tciSendConnectedMC (
            in TriPortIdType sender,
            in TriComponentIdListType receivers,
            in Value sendMessage
            );

  void tciCallConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList
            );
  void tciCallConnectedBC (
            in TriPortIdType sender,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList
            );
  void tciCallConnectedMC (
            in TriPortIdType sender,
            in TriComponentIdListType receivers,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList
            );

  void tciReplyConnected (
            in TriPortIdType sender,
            in TriComponentIdType receiver,
            in TriSignatureIdType signature,
            in TciParameterListType parameterList,
            in Value returnValue
            );
  void tciReplyConnectedBC (
            in TriPortIdType sender,
            in TriSignatureIdType signature,
```

```
              in TciParameterListType parameterList,
              in Value returnValue
              );
      void tciReplyConnectedMC (
              in TriPortIdType sender,
              in TriComponentIdListType receivers,
              in TriSignatureIdType signature,
              in TciParameterListType parameterList,
              in Value returnValue
              );

      void tciRaiseConnected (
              in TriPortIdType        sender,
              in TriComponentIdType receiver,
              in TriSignatureIdType signature,
              in Value except
              );
      void tciRaiseConnectedBC (
              in TriPortIdType sender,
              in TriSignatureIdType signature,
              in Value except
              );
      void tciRaiseConnectedMC (
              in TriPortIdType sender,
              in TriComponentIdListType receivers,
              in TriSignatureIdType signature,
              in Value except
              );

      TriComponentIdType tciCreateTestComponentReq (
              in TciTestComponentKindType kind,
              in Type componentType,
              in TString name
              );
      void tciStartTestComponentReq (
              in TriComponentIdType comp,
              in TciBehaviourIdType behavior,
              in TciParameterListType parameterList
              );
      void tciStopTestComponentReq (
              in TriComponentIdType comp
              );
      void tciConnectReq (
              in TriPortIdType fromPort,
              in TriPortIdType toPort
              );
      void tciDisconnectReq (
              in TriPortIdType fromPort,
              in TriPortIdType toPort
              );
      void tciTestComponentTerminatedReq (
              in TriComponentIdType comp,
              in VerdictValue verdict
              );
      TBoolean tciTestComponentRunningReq (
              in TriComponentIdType comp
              );
      TriComponentIdType tciGetMTCReq ();
      void tciMapReq (
              in TriPortIdType fromPort,
              in TriPortIdType toPort
              );
      void tciUnmapReq (
              in TriPortIdType fromPort,
              in TriPortIdType toPort
              );
      void tciExecuteTestCaseReq (
              in TciTestCaseIdType testCaseId,
              in TriPortIdListType tsiPortList
              );
      void tciResetReq ();
      TBoolean tciTestComponentDoneReq (
              in TriComponentIdType comp
              );
};

// *************************************************************************
// Test Logging Interface
// -  Provided
// *************************************************************************
```

```
interface TCI_TL_Provided {
  void tliTcExecute(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciTestCaseIdType tcId,
        in TriParameterListType triPars, in TriTimerDurationType dur
        );
  void tliTcStart(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciTestCaseIdType tcId,
        in TciParameterListType tciPars, in TriTimerDurationType dur
        );
  void tliTcStop(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
  void tliTcStarted(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciTestCaseIdType tcId,
        in TciParameterListType tciPars, in TriTimerDurationType dur
        );
  void tliTcTerminated(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciTestCaseIdType tcId,
        in TciParameterListType tciPars, in VerdictValue verdict);

  void tliCtrlStart(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
  void tliCtrlStop(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
  void tliCtrlTerminated(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c);

  void tliMSend_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
        in TriAddressType address, in TciStatusType encoderFailure,
        in TriMessageType msg, in TriStatusType transmissionFailure
        );
  void tliMSend_m_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
        in TciStatusType encoderFailure, in TriMessageType msg,
        in TriStatusType transmissionFailure
        );
  void tliMSend_m_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
        in TriAddressListType addresses, in TciStatusType encoderFailure,
        in TriMessageType msg, in TriStatusType transmissionFailure
        );

  void tliMSend_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
        in TriStatusType transmissionFailure
        );
  void tliMSend_c_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in Value msgValue,
        in TriStatusType transmissionFailure
        );
  void tliMSend_c_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to, in Value msgValue,
        in TriStatusType transmissionFailure);

  void tliMDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from, in TriMessageType
        msg,
        in TriAddressType address
        );
  void tliMDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from, in Value msgValue
        );
    void tliMMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
        in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs,
         in TriAddressType address, in TciValueTemplate addressTmpl
        );
    void tliMMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
        in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs,
         in TriComponentIdType from, in TciNonValueTemplate fromTmpl
        );
    void tliMReceive_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
        in TciValueTemplate msgTmpl, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
    void tliMReceive_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
        in TciValueTemplate msgTmpl, in TriComponentIdType fromComp,
        in TciNonValueTemplate fromTmpl
        );

    void tliPrCall_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TriAddressType address, in TciStatusType encoderFailure,
        in TriParameterListType triPars, in TriStatusType transmissionFailure
        );
     void tliPrCall_m_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TciStatusType encoderFailure, in TriParameterListType triPars,
        in TriStatusType transmissionFailure
        );
     void tliPrCall_m_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TriAddressListType addresses, in TciStatusType encoderFailure,
        in TriParameterListType triPars,  in TriStatusType transmissionFailure
        );

    void tliPrCall_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TriStatusType transmissionFailure
        );
     void tliPrCall_c_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TriStatusType transmissionFailure
        );
       void tliPrCall_c_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
        in TriSignatureIdType signature, in TciParameterListType tciPars,
        in TriStatusType transmissionFailure
        );

    void tliPrGetCallDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
        in TriSignatureIdType signature, in TriParameterListType triPars,
        in TriAddressType address
        );
    void tliPrGetCallDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
        in TriSignatureIdType signature, in TciParameterListType tciPars
        );
```

```
void tliPrGetCallMismatch_m(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs,
      in TriAddressType address, in TciValueTemplate addressTmpl
      );
void tliPrGetCallMismatch_c(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs,
      in TriComponentIdType from, in TciNonValueTemplate fromTmpl
      );
void tliPrGetCall_m(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in TciValueTemplate parsTmpl, in TriAddressType address,
      in TciValueTemplate addressTmpl
      );
void tliPrGetCall_c(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in TciValueTemplate parsTmpl, in TriComponentIdType from,
      in TciNonValueTemplate fromTmpl
      );

void tliPrReply_m(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in Value replValue, in TriAddressType address,
      in TciStatusType encoderFailure, TriParameterList triPars, in TriParameterType repl,
      in TriStatusType transmissionFailure
      );
 void tliPrReply_m_BC(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
      in TriSignatureIdType signature, in TciParameterListType tciPars, in Value replValue,
      in TciStatusType encoderFailure, TriParameterList triPars, in TriParameterType repl,
      in TriStatusType transmissionFailure
      );
 void tliPrReply_m_MC(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
      in TriSignatureIdType signature, in TciParameterListType tciPars, in Value replValue,
      in TriAddressListType addresses, in TciStatusType encoderFailure,
      TriParameterList triPars, in TriParameterType repl, in TriStatusType transmissionFailure
      );

void tliPrReply_c(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
      in TriSignatureIdType signature, in TciParameterListType tciPars,
      in Value replValue, in TriStatusType transmissionFailure
      );
 void tliPrReply_c_BC(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
      in TriSignatureIdType signature, in Value parsValue, in Value replValue,
      in TriStatusType transmissionFailure
      );
 void tliPrReply_c_MC(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
      in TriSignatureIdType signature, in Value parsValue, in Value replValue,
      in TriStatusType transmissionFailure
      );

void tliPrGetReplyDetected_m(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
      in TriSignatureIdType signature, in TriParameterListType triPars,
      in TriParameterType repl, in TriAddressType address
      );
void tliPrGetReplyDetected_c(
      in TString am, in TInteger ts, in TString src, in TInteger line,
      in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
```

```
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value replValue
            );
    void tliPrGetReplyMismatch_m(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at,
            in TriSignatureIdType signature,
            in TciParameterListType tciPars, in TciValueTemplate parsTmpl,
            in Value replValue, in TciValueTemplate replyTmpl,
            in TciValueDifferenceList diffs, in TriAddressType address,
            in TciValueTemplate addressTmpl
            );
    void tliPrGetReplyMismatch_c(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at,
            in TriSignatureIdType signature,
            in TciParameterListType tciPars, in TciValueTemplate parsTmpl,
            in Value replValue, in TciValueTemplate replyTmpl,
            in TciValueDifferenceList diffs, in TriComponentIdType from,
            in TciNonValueTemplate fromTmpl
            );
    void tliPrGetReply_m(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at,
            in TriSignatureIdType signature,
            in TciParameterListType tciPars, in TciValueTemplate parsTmpl,
            in Value replValue, in TciValueTemplate replyTmpl,
            in TriAddressType address, in TciValueTemplate addressTmpl
            );
    void tliPrGetReply_c(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at,
            in TriSignatureIdType signature,
            in TciParameterListType tciPars, in TciValueTemplate parsTmpl,
            in Value replValue, in TciValueTemplate replyTmpl,
            in TriComponentIdType from, in TciNonValueTemplate fromTmpl
            );

    void tliPrRaise_m(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TriAddressType address, in TciStatusType encoderFailure,
            in TriExceptionType exc, in TriStatusType transmissionFailure
            );
     void tliPrRaise_m_BC(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TciStatusType encoderFailure, in TriExceptionType exc,
            in TriStatusType transmissionFailure
            );
     void tliPrRaise_m_MC(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TriAddressListType addresses,
            in TciStatusType encoderFailure, in TriExceptionType exc,
            in TriStatusType transmissionFailure
            );

    void tliPrRaise_c(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TriStatusType transmissionFailure
            );
     void tliPrRaise_c_BC(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TriStatusType transmissionFailure
            );
     void tliPrRaise_c_MC(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType at, in TriPortIdListType to,
            in TriSignatureIdType signature, in TciParameterListType tciPars,
            in Value excValue, in TriStatusType transmissionFailure
            );
```

```
void tliPrCatchDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
        in TriSignatureIdType signature,
        in TriExceptionType exc, in TriAddressType address
        );
void tliPrCatchDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
        in TriSignatureIdType signature, in Value excValue
        );
void tliPrCatchMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
void tliPrCatchMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );
void tliPrCatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature,
        in Value excValue, in TciValueTemplate excTmpl,
        in TriAddressType address, in TciValueTemplate addressTmpl);

void tliPrCatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature,
        in Value excValue, in TciValueTemplate excTmpl,
        in TriComponentIdType from, in TciNonValueTemplate fromTmpl
        );
void tliPrCatchTimeoutDetected(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature
        );
void tliPrCatchTimeout(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType at,
        in TriSignatureIdType signature
        );
void tliCCreate(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriComponentIdType comp,
        in TString name, in TBoolean alive
        );
void tliCStart(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriComponentIdType comp,
        in TciBehaviourIdType name, in TciParameterListType tciPars
        );
void tliCRunning(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriComponentIdType comp, in ComponentStatusType status
        );
 void tliCAlive(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c,
         in TriComponentIdType comp, in ComponentStatusType status
         );
 void tliCStop(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCKill(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCDoneMismatch(
         in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
                in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl
                );
    void tliCKilledMismatch(
                in TString am, in TInteger ts, in TString src, in TInteger line,
                in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl
                );
    void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line,
                in TriComponentIdType c, in TciNonValueTemplate compTmpl
                );
    void tliCKilled(
                in TString am, in TInteger ts, in TString src, in TInteger line,
                in TriComponentIdType c, in TciNonValueTemplate compTmpl
                );
void tliCTerminated(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in VerdictValue verdict
            );
void tliPConnect(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2
            );
void tliPDisconnect(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
            in TriComponentIdType c2, in TriPortIdType port2
            );
void tliPMap(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2
            );
void tliPUnmap(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
            in TriComponentIdType c2, in TriPortIdType port2
            );
void tliPClear(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port
            );
void tliPStart(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port
            );
void tliPStop(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port
            );
void tliPHalt(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriPortIdType port
            );
void tliEncode(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in Value val, in TciStatusType encoderFailure,
            in TriMessageType msg, in TString codec
            );
void tliDecode(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriMessageType msg,
            in TciStatusType decoderFailure, in Value val, in TString codec
            );
void tliTTimeoutDetected(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriTimerIdType timer
            );
void tliTTimeoutMismatch(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriTimerIdType timer, in TciNonValueTemplate timerTmpl
            );
void tliTTimeout(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriTimerIdType timer, in TciNonValueTemplate timerTmpl
            );
void tliTStart(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TriTimerIdType timer,
            in TriTimerDurationType dur
            );
void tliTStop(
            in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
        in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur
        );
void tliTRead(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer,
        in TriTimerDurationType elapsed
        );
void tliTRunning(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer, in TimerStatusType status
        );
void tliSEnter(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars,
        in TString kind
        );
void tliSLeave(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars,
        in Value returnValue, in TString kind
        );
void tliVar(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in QualifiedName name, in Value varValue
        );
void tliModulePar(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in QualifiedName name, in Value parValue
        );
void tliGetVerdict(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in VerdictValue verdict
        );
void tliSetVerdict(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in VerdictValue verdict
        );
void tliLog(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString log
        );
void tliAEnter(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliALeave(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliADefaults(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliAActivate(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars,
        in Value ref
        );
void tliADeactivate(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in Value ref
        );
void tliANomatch(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliARepeat(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliAWait(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c
        );
void tliAction(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString action
    );
void tliMatch(
        in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
                in TriComponentIdType c, in Value expr, in TciValueTemplate tmpl
        );
    void tliMatchMismatch(
            in TString am, in TInteger ts, in TString src,
            in TInteger line, in TriComponentIdType c, in Value expr,
            in TciValueTemplate tmpl, in TciValueDifferenceList diffs
            );
    void tliInfo(
            in TString am, in TInteger ts, in TString src,
            in TInteger line, in TriComponentIdType c,
            in TInteger level, in TString info
            );
    };
};
```

Annex B

XML Mapping for TCI-TL Provided

(This annex forms an integral part of this Recommendation)

This annex defines a mapping for the logging interface of TCI using eXtended Markup Language (XML) schema definitions.

B.1   TCI-TL XML Schema for Simple Types

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
            xmlns:SimpleTypes="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
            elementFormDefault="qualified">

    <!-- Basic definitions -->
    <xsd:simpleType name="xpath">
        <!-- this string should be XPATH compliant -->
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TBoolean">
        <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>

    <xsd:simpleType name="TString">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TInteger">
        <xsd:restriction base="xsd:integer"/>
    </xsd:simpleType>

    <!-- Miscellaneous -->
    <xsd:simpleType name="TriTimerDurationType">
        <xsd:restriction base="xsd:float"/>
    </xsd:simpleType>

    <xsd:simpleType name="TciParameterPassingModeType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="in"/>
            <xsd:enumeration value="inout"/>
            <xsd:enumeration value="out"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TriStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TRI_Ok"/>
            <xsd:enumeration value="TRI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TciStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TCI_Ok"/>
            <xsd:enumeration value="TCI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="ComponentStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="inactiveC"/>
            <xsd:enumeration value="runningC"/>
            <xsd:enumeration value="stoppedC"/>
            <xsd:enumeration value="killedC"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TimerStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="runningT"/>
```

```
            <xsd:enumeration value="inactiveT"/>
            <xsd:enumeration value="expiredT"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="PortStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="startedP"/>
            <xsd:enumeration value="haltedP"/>
            <xsd:enumeration value="stoppedP"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

B.2   TCI-TL XML Schema for Types

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
    xmlns:Types="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
    xmlns:SimpleTypes="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
    xmlns:Values="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
    xmlns:Templates="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
        elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
     schemaLocation="Values_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
     schemaLocation="SimpleTypes_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
     schemaLocation="Templates_v3_2_1.xsd"/>

    <!-- Connection -->
    <xsd:complexType name="TriPortIdType">
        <xsd:sequence>
            <xsd:element name="comp" type="Types:TriComponentIdType" />
            <xsd:element name="port" type="Types:Port" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TriPortIdListType">
        <xsd:sequence>
            <xsd:element name="comp" type="Types:TriPortIdType" minOccurs="0"
             maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Port">
        <xsd:sequence>
            <xsd:element name="id" type="Types:Id"  />
            <xsd:element name="index" type="xsd:int" minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TriComponentIdType">
        <xsd:sequence>
            <xsd:choice>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="id" type="Types:Id"  />
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TriComponentIdListType">
        <xsd:sequence>
            <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="0"
             maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <!-- Communication -->
    <xsd:complexType name="TriMessageType">
        <xsd:attribute name="val" type="xsd:hexBinary"/>
    </xsd:complexType>

    <xsd:complexType name="TriParameterType">
        <xsd:sequence>
            <xsd:element name="val" type="xsd:hexBinary"  />
```

```xml
        </xsd:sequence>
        <xsd:attribute name="name" type="SimpleTypes:TString"/>
        <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
    </xsd:complexType>

    <xsd:complexType name="TriParameterListType">
        <xsd:sequence>
            <xsd:element name="par" type="Types:TriParameterType" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TriExceptionType">
        <xsd:attribute name="val" type="SimpleTypes:TString"/>
    </xsd:complexType>

    <xsd:complexType name="TciValueListType">
        <xsd:complexContent>
            <xsd:extension base="Values:RecordValue"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="TriSignatureIdType">
        <xsd:attribute name="val" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="TriAddressType">
        <xsd:attribute name="val" type="SimpleTypes:TString"/>
    </xsd:complexType>

    <xsd:complexType name="TriAddressListType">
        <xsd:sequence>
            <xsd:element name="addr" type="Types:TriAddressType" minOccurs="0"
             maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>


    <!-- Miscellaneous -->
    <xsd:complexType name="TriTimerIdType">
        <xsd:sequence>
            <xsd:element name="id" type="Types:Id"  />
        </xsd:sequence>
    </xsd:complexType>

    <!-- Basic definitions -->
    <xsd:complexType name="QualifiedName">
        <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- general TCI abstract data types -->
    <xsd:complexType name="TciBehaviourIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName"  />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TciTestCaseIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName"  />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TciParameterType">
        <xsd:sequence>
            <xsd:element name="val" type="Values:Value"  />
        </xsd:sequence>
        <xsd:attribute name="name" type="SimpleTypes:TString"/>
        <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
    </xsd:complexType>

    <xsd:complexType name="TciParameterListType">
        <xsd:sequence>
            <xsd:element name="par" type="Types:TciParameterType" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <!-- general identifier structure for test components, ports and timer -->
```

```
<xsd:complexType name="Id">
    <xsd:sequence>
        <xsd:element name="name" type="SimpleTypes:TString"  />
        <xsd:element name="id" type="SimpleTypes:TString"  />
        <xsd:element name="type" type="SimpleTypes:TString"  />
    </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

B.3   TCI-TL XML Schema for Values

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
    xmlns:Values="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
    xmlns:Templates="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
    xmlns:SimpleTypes="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
     schemaLocation="Templates_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
     schemaLocation="SimpleTypes_v3_2_1.xsd"/>

    <xsd:attributeGroup name="ValueAtts">
        <xsd:attribute name="name" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="type" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="module" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="annotation" type="SimpleTypes:TString" use="optional"/>
    </xsd:attributeGroup>

    <xsd:complexType name="Value" mixed="true">
        <xsd:choice>
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"/>
            <xsd:element name="charstring" type="Values:CharstringValue"/>
            <xsd:element name="universal_charstring" type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
                </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <!-- general event elements -->
    <xsd:complexType name="IntegerValue">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="null" type="Templates:null"/>
            <xsd:element name="omit" type="Templates:omit"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="FloatValue">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="null" type="Templates:null"/>
            <xsd:element name="omit" type="Templates:omit"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="BooleanValue">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="null" type="Templates:null"/>
```

```
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="ObjidValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="VerdictValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="BitstringValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="HexstringValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="OctetstringValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="CharstringValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="UniversalCharstringValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
                <xsd:element name="omit" type="Templates:omit"/>
            </xsd:choice>
            <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:complexType>

        <xsd:complexType name="RecordValue">
            <xsd:choice>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Values:IntegerValue"/>
                    <xsd:element name="float" type="Values:FloatValue"/>
                    <xsd:element name="boolean" type="Values:BooleanValue"/>
                    <xsd:element name="objid" type="Values:ObjidValue"/>
                    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
                    <xsd:element name="bitstring" type="Values:BitstringValue"/>
                    <xsd:element name="hexstring" type="Values:HexstringValue"/>
                    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
                    <xsd:element name="charstring" type="Values:CharstringValue"/>
```

```
                <xsd:element name="universal_charstring"
                    type="Values:UniversalCharstringValue"/>
                <xsd:element name="record" type="Values:RecordValue"/>
                <xsd:element name="record_of" type="Values:RecordOfValue"/>
                <xsd:element name="set" type="Values:SetValue"/>
                <xsd:element name="set_of" type="Values:SetOfValue"/>
                <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
                <xsd:element name="union" type="Values:UnionValue"/>
                <xsd:element name="anytype" type="Values:AnytypeValue"/>
                <xsd:element name="address" type="Values:AddressValue"/>
            </xsd:choice>
            <xsd:element name="null" type="Templates:null"/>
            <xsd:element name="omit" type="Templates:omit"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="RecordOfValue">
        <xsd:choice>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="null" type="Templates:null"/>
            <xsd:element name="omit" type="Templates:omit"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="SetValue">
        <xsd:choice>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="integer" type="Values:IntegerValue"/>
                <xsd:element name="float" type="Values:FloatValue"/>
                <xsd:element name="boolean" type="Values:BooleanValue"/>
                <xsd:element name="objid" type="Values:ObjidValue"/>
                <xsd:element name="verdicttype" type="Values:VerdictValue"/>
                <xsd:element name="bitstring" type="Values:BitstringValue"/>
                <xsd:element name="hexstring" type="Values:HexstringValue"/>
                <xsd:element name="octetstring" type="Values:OctetstringValue"/>
                <xsd:element name="charstring" type="Values:CharstringValue"/>
                <xsd:element name="universal_charstring"
                    type="Values:UniversalCharstringValue"/>
                <xsd:element name="record" type="Values:RecordValue"/>
                <xsd:element name="record_of" type="Values:RecordOfValue"/>
                <xsd:element name="set" type="Values:SetValue"/>
                <xsd:element name="set_of" type="Values:SetOfValue"/>
                <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
                <xsd:element name="union" type="Values:UnionValue"/>
```

```
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
        </xsd:choice>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="SetOfValue">
    <xsd:choice>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="EnumeratedValue">
    <xsd:choice>
        <xsd:element name="value" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="omit" type="Templates:omit"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="UnionValue">
    <xsd:choice>
        <xsd:element name="integer" type="Values:IntegerValue"/>
        <xsd:element name="float" type="Values:FloatValue"/>
        <xsd:element name="boolean" type="Values:BooleanValue"/>
        <xsd:element name="objid" type="Values:ObjidValue"/>
        <xsd:element name="verdicttype" type="Values:VerdictValue"/>
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:CharstringValue"/>
        <xsd:element name="universal_charstring"
            type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
```

```
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="AnytypeValue">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"/>
            <xsd:element name="charstring" type="Values:OctetstringValue"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="address" type="Values:AddressValue"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="AddressValue">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
            <xsd:element name="bitstring" type="Values:BitstringValue"/>
            <xsd:element name="hexstring" type="Values:HexstringValue"/>
            <xsd:element name="octetstring" type="Values:OctetstringValue"/>
            <xsd:element name="charstring" type="Values:OctetstringValue"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>
</xsd:schema>
```

B.4   TCI-TL XML Schema for Templates

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
    xmlns:Templates="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
    xmlns:Values="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
    xmlns:Types="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
    xmlns:SimpleTypes="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
     schemaLocation="Values_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
schemaLocation="Types_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
     schemaLocation="SimpleTypes_v3_2_1.xsd"/>

    <xsd:complexType name="TciValueTemplate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Values:Value">
            <xsd:choice>
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
```

```
            <xsd:element name="float" type="Templates:FloatTemplate"/>
            <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
            <xsd:element name="objid" type="Templates:ObjidTemplate"/>
            <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
            <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
            <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
            <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
            <xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
            <xsd:element name="union" type="Templates:UnionTemplate"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
            <xsd:element name="address" type="Templates:AddressTemplate"/>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        </xsd:choice>
         </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="omit">
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="any">
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="anyoromit">
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>


    <xsd:complexType name="TciNonValueTemplate">
        <xsd:sequence>
            <xsd:choice>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="all" type="Templates:all"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="all">
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="null">
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="TciValueDifference">
        <xsd:sequence>
            <xsd:element name="val" type="SimpleTypes:xpath"/>
            <xsd:element name="tmpl" type="SimpleTypes:xpath"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
        <xsd:attribute name="desc" type="SimpleTypes:TString" use="optional"/>
    </xsd:complexType>

    <xsd:complexType name="TciValueDifferenceList">
            <xsd:sequence>
                    <xsd:element name="diff" type="Templates:TciValueDifference"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>

    <xsd:complexType name="IntegerTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
```

```
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="FloatTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="BooleanTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="ObjidTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="BitstringTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="HexstringTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="OctetstringTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>

    <xsd:complexType name="CharstringTemplate">
        <xsd:choice>
```

```
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>


    <xsd:complexType name="UniversalCharstringTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>


    <xsd:complexType name="VerdictTemplate">
        <xsd:choice>
            <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="Templates:null"/>
            </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>


    <xsd:complexType name="RecordTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:RecordValue">
                <xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                        <xsd:element name="float" type="Templates:FloatTemplate"/>
                        <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                        <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                        <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                        <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                        <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                        <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                        <xsd:element name="universal_charstring"
                            type="Templates:UniversalCharstringTemplate"/>
                        <xsd:element name="record" type="Templates:RecordTemplate"/>
                        <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                        <xsd:element name="set" type="Templates:SetTemplate"/>
                        <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                        <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                        <xsd:element name="union" type="Templates:UnionTemplate"/>
                        <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                        <xsd:element name="address" type="Templates:AddressTemplate"/>
                    </xsd:choice>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="null" type="Templates:null"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>


    <xsd:complexType name="RecordOfTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:RecordOfValue">
                <xsd:choice>
                    <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
```

```
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:SetValue">
            <xsd:choice>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                    <xsd:element name="record" type="Templates:RecordTemplate"/>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                    <xsd:element name="set" type="Templates:SetTemplate"/>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                    <xsd:element name="union" type="Templates:UnionTemplate"/>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                    <xsd:element name="address" type="Templates:AddressTemplate"/>
                </xsd:choice>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                <xsd:element name="null" type="Templates:null"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetOfTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:SetOfValue">
            <xsd:choice>
                <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
```

```
                maxOccurs="unbounded"/>
            <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="charstring" type="Templates:CharstringTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="universal_charstring"
                type="Templates:UniversalCharstringTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            <xsd:element name="null" type="Templates:null"/>
        </xsd:choice>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="EnumeratedTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:EnumeratedValue">
            <xsd:choice>
                <xsd:element name="value" type="SimpleTypes:TString"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="Templates:null"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="UnionTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:UnionValue">
            <xsd:choice>
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                <xsd:element name="float" type="Templates:FloatTemplate"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                <xsd:element name="universal_charstring"
                    type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                <xsd:element name="omit" type="Templates:omit"/>
                <xsd:element name="any" type="Templates:any"/>
                <xsd:element name="anyoromit" type="Templates:anyoromit"/>
```

```
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="null" type="Templates:null"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AnytypeTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                    <xsd:element name="record" type="Templates:RecordTemplate"/>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                    <xsd:element name="set" type="Templates:SetTemplate"/>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                    <xsd:element name="union" type="Templates:UnionTemplate"/>
                    <xsd:element name="address" type="Templates:AddressTemplate"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="null" type="Templates:null"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AddressTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                    <xsd:element name="record" type="Templates:RecordTemplate"/>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                    <xsd:element name="set" type="Templates:SetTemplate"/>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                    <xsd:element name="union" type="Templates:UnionTemplate"/>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="null" type="Templates:null"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

B.5   TCI-TL XML Schema for Events

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/TTCN-3/tci/Events_v3_2_1.xsd"
    xmlns:Events="http://uri.etsi.org/TTCN-3/tci/Events_v3_2_1.xsd"
    xmlns:Types="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
    xmlns:Templates="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
    xmlns:SimpleTypes="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
```

```
        xmlns:Values="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
        elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/SimpleTypes_v3_2_1.xsd"
     schemaLocation="SimpleTypes_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
schemaLocation="Types_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
     schemaLocation="Values_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Templates_v3_2_1.xsd"
     schemaLocation="Templates_v3_2_1.xsd"/>

    <!-- common definition for all events -->
    <xsd:complexType name="Event" mixed="true">
        <xsd:sequence>
            <xsd:element name="am" type="SimpleTypes:TString"/>
        </xsd:sequence>
        <xsd:attribute name="ts" type="xsd:long" use="required"/>
        <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

        <!-- general identifier structure for test components, ports and timer -->
        <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="id" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- this event is extended by all port configuration events -->
    <xsd:complexType name="PortConfiguration">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port1" type="Types:TriPortIdType"  />
                    <xsd:element name="port2" type="Types:TriPortIdType"  />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- this event is extended by all port status events -->
    <xsd:complexType name="PortStatus">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="stat" type="SimpleTypes:PortStatusType"  minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- testcases -->
    <xsd:complexType name="tliTcExecute">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="triPars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcStart">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcStop">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
```

```
        </xsd:complexType>

        <xsd:complexType name="tliTcStarted">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTcTerminated">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                        <xsd:element name="verdict" type="Values:VerdictValue"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- control -->
        <xsd:complexType name="tliCtrlStart">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliCtrlStop">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliCtrlTerminated">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- asynchronous communication -->
        <xsd:complexType name="tliMSend_m">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="at" type="Types:TriPortIdType"/>
                        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                        <xsd:element name="msgValue" type="Values:Value"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:choice>
                            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                            <xsd:sequence>
                                <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
                                <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                            </xsd:sequence>
                        </xsd:choice>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliMSend_m_BC">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="at" type="Types:TriPortIdType"/>
                        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                        <xsd:element name="msgValue" type="Values:Value"/>
                        <xsd:choice>
                            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                            <xsd:sequence>
                                <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
```

```
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="msgValue" type="Types:TriMessageType"/>
```

```
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliMDetected_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="msgValue" type="Values:Value"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliMMismatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliMMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliMReceive_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliMReceive_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                <xsd:element name="fromComp" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
```

```
            </xsd:complexType>

            <!-- synchronous communication -->
            <xsd:complexType name="tliPrCall_m">
                <xsd:complexContent mixed="true">
                    <xsd:extension base="Events:Event">
                        <xsd:sequence>
                            <xsd:element name="at" type="Types:TriPortIdType"/>
                            <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                            <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                            <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                            <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                            <xsd:choice>
                                <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                                <xsd:sequence>
                                    <xsd:element name="triPars" type="Types:TriParameterListType"/>
                                    <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                                </xsd:sequence>
                            </xsd:choice>
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="tliPrCall_m_BC">
                <xsd:complexContent mixed="true">
                    <xsd:extension base="Events:Event">
                        <xsd:sequence>
                            <xsd:element name="at" type="Types:TriPortIdType"/>
                            <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                            <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                            <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                            <xsd:choice>
                                <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                                <xsd:sequence>
                                    <xsd:element name="triPars" type="Types:TriParameterListType"/>
                                    <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                                </xsd:sequence>
                            </xsd:choice>
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="tliPrCall_m_MC">
                <xsd:complexContent mixed="true">
                    <xsd:extension base="Events:Event">
                        <xsd:sequence>
                            <xsd:element name="at" type="Types:TriPortIdType"/>
                            <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                            <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                            <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                            <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                            <xsd:choice>
                                <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                                <xsd:sequence>
                                    <xsd:element name="triPars" type="Types:TriParameterListType"/>
                                    <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                                </xsd:sequence>
                            </xsd:choice>
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="tliPrCall_c">
                <xsd:complexContent mixed="true">
                    <xsd:extension base="Events:Event">
                        <xsd:sequence>
                            <xsd:element name="at" type="Types:TriPortIdType"/>
                            <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                            <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                            <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                            <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
```

```
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetCallDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="triPars" type="Types:TriParameterListType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetCallDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetCallMismatch_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```xsd
    <xsd:complexType name="tliPrGetCallMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="fromTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetCall_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetCall_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
                            minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="triPars" type="Types:TriParameterListType"/>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
                                    type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
```

```
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
                            minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="triPars" type="Types:TriParameterListType"/>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
                                type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
                            minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="triPars" type="Types:TriParameterListType"/>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
                                type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
```

284   **Rec. ITU-T Z.166 (11/2007)**

```
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="triPars" type="Types:TriParameterListType"/>
                    <xsd:element name="repl" type="Types:TriParameterType"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyMismatch_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReply_m">
        <xsd:complexContent mixed="true">
```

```xsd
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="at" type="Types:TriPortIdType"/>
                        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                        <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="replValue" type="Values:Value"/>
                        <xsd:element name="replTmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

    <xsd:complexType name="tliPrGetReply_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
    <xsd:complexType name="tliPrRaise_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCatchDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
```

```
                <xsd:element name="exc" type="Types:TriExceptionType"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
```

```
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCatchTimeoutDetected">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCatchTimeout">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="at" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- components -->
    <xsd:complexType name="tliCCreate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="name" type="SimpleTypes:TString"/>
                    <xsd:element name="alive" type="SimpleTypes:TBoolean"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCStart">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="name" type="Types:TciBehaviourIdType"/>
                    <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCRunning">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>


    <xsd:complexType name="tliCAlive">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCStop">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
```

```xml
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCKill">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCDoneMismatch">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCKilledMismatch">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="comp" type="Types:TriComponentIdType"/>
                    <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCDone">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCKilled">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCTerminated">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="verdict" type="Values:VerdictValue" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- ports -->
    <xsd:complexType name="tliPConnect">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPDisconnect">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>
```

```xml
<xsd:complexType name="tliPMap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPUnmap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPClear">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortStatus"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStart">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortStatus"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStop">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortStatus"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPHalt">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortStatus"/>
    </xsd:complexContent>
</xsd:complexType>

<!-- codec -->
<xsd:complexType name="tliEncode">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="val" type="Values:Value"/>
                <xsd:choice>
                    <xsd:element name="msg" type="Types:TriMessageType"/>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                </xsd:choice>
                <xsd:element name="codec" type="SimpleTypes:TString"
                    minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliDecode" mixed="true">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:choice>
                    <xsd:element name="val" type="Values:Value"/>
                    <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                </xsd:choice>
                <xsd:element name="msg" type="Types:TriMessageType"/>
                <xsd:element name="codec" type="SimpleTypes:TString"
                    minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- timers -->
<xsd:complexType name="tliTTimeoutDetected">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="timer" type="Types:TriTimerIdType"  />
            </xsd:sequence>
        </xsd:extension>
```

```
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTTimeoutMismatch">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"  />
                        <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate"  />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTTimeout">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"  />
                        <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate"  />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTStart">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"/>
                        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTStop">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"/>
                        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTRead">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"/>
                        <xsd:element name="elapsed" type="SimpleTypes:TriTimerDurationType"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliTRunning">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="timer" type="Types:TriTimerIdType"/>
                    </xsd:sequence>
                    <xsd:attribute name="status" type="SimpleTypes:TimerStatusType"/>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- scope -->
        <xsd:complexType name="tliSEnter">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="name" type="Types:QualifiedName"  />
                        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                        <xsd:element name="kind" type="SimpleTypes:TString"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
```

```xml
        </xsd:complexType>

        <xsd:complexType name="tliSLeave">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="name" type="Types:QualifiedName"  />
                        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                        <xsd:element name="returnValue" type="Values:Value" minOccurs="0"/>
                        <xsd:element name="kind" type="SimpleTypes:TString"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- variables and module parameter -->
        <xsd:complexType name="tliVar">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="name" type="Types:QualifiedName"  />
                        <xsd:element name="val" type="Values:Value" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliModulePar">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="name" type="Types:QualifiedName"  />
                        <xsd:element name="val" type="Values:Value" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- verdicts -->
        <xsd:complexType name="tliGetVerdict">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="verdict" type="Values:VerdictValue"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliSetVerdict">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="verdict" type="Values:VerdictValue"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- log -->
        <xsd:complexType name="tliLog">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="log" type="SimpleTypes:TString"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <!-- alt -->
        <xsd:complexType name="tliAEnter">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliALeave">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
```

```
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliADefaults">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliAActivate">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="name" type="Types:QualifiedName"  />
                        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                        <xsd:element name="ref" type="Values:Value"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliADeactivate">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="ref" type="Values:Value"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliANomatch">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliARepeat">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliAWait">
            <xsd:complexContent>
                <xsd:extension base="Events:Event"/>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliAction">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="action" type="SimpleTypes:TString"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliMatch">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="expr" type="Values:Value"/>
                        <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="tliMatchMismatch">
            <xsd:complexContent mixed="true">
                <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="expr" type="Values:Value"/>
                        <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
```

```
        </xsd:complexType>

    <xsd:complexType name="tliInfo">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="level" type="SimpleTypes:TInteger"/>
                    <xsd:element name="info" type="SimpleTypes:TString"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

B.6   TCI-TL XML Schema for a Log

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/TTCN-3/tci/TLI_v3_2_1.xsd"
    xmlns:TLI="http://uri.etsi.org/TTCN-3/tci/TLI_v3_2_1.xsd"
    xmlns:Types="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
    xmlns:Values="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
    xmlns:Events="http://uri.etsi.org/TTCN-3/tci/Events_v3_2_1.xsd"
    elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Types_v3_2_1.xsd"
schemaLocation="Types_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Values_v3_2_1.xsd"
     schemaLocation="Values_v3_2_1.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/TTCN-3/tci/Events_v3_2_1.xsd"
     schemaLocation="Events_v3_2_1.xsd"/>

    <xsd:element name="logfile" type="TLI:LogModule"/>
    <xsd:complexType name="LogModule">
        <xsd:sequence>
            <xsd:element name="header" type="TLI:Header"/>
            <xsd:element name="body" type="TLI:Body"/>
            <xsd:element name="trailer" type="TLI:Trailer" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Header">
        <xsd:sequence>
          <!-- logging version -->
          <xsd:element name="version" type="xsd:string"/>
        <!-- begin of the log -->
            <xsd:element name="ts" type="xsd:long"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Trailer">
        <xsd:choice>
            <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="Body">
        <xsd:choice maxOccurs="unbounded">

            <!-- test cases operations -->
            <xsd:element name="tliTcExecute" type="Events:tliTcExecute"/>
            <xsd:element name="tliTcStart" type="Events:tliTcStart"/>
            <xsd:element name="tliTcStop" type="Events:tliTcStop"/>
            <xsd:element name="tliTcStarted" type="Events:tliTcStarted"/>
            <xsd:element name="tliTcTerminated" type="Events:tliTcTerminated"/>

            <!-- control operations -->
            <xsd:element name="tliCtrlStart" type="Events:tliCtrlStart"/>
            <xsd:element name="tliCtrlStop" type="Events:tliCtrlStop"/>
            <xsd:element name="tliCtrlTerminated" type="Events:tliCtrlTerminated"/>

            <!-- asynchronous communication -->
            <xsd:element name="tliMSend_m" type="Events:tliMSend_m"/>
            <xsd:element name="tliMSend_c" type="Events:tliMSend_c"/>
            <xsd:element name="tliMSend_m_BC" type="Events:tliMSend_m_BC"/>
            <xsd:element name="tliMSend_c_BC" type="Events:tliMSend_c_BC"/>
            <xsd:element name="tliMSend_m_MC" type="Events:tliMSend_m_MC"/>
            <xsd:element name="tliMSend_c_MC" type="Events:tliMSend_c_MC"/>
            <xsd:element name="tliMDetected_m" type="Events:tliMDetected_m"/>
            <xsd:element name="tliMDetected_c" type="Events:tliMDetected_c"/>
            <xsd:element name="tliMMismatch_m" type="Events:tliMMismatch_m"/>
```

```xml
<xsd:element name="tliMMismatch_c" type="Events:tliMMismatch_c"/>
<xsd:element name="tliMReceive_m" type="Events:tliMReceive_m"/>
<xsd:element name="tliMReceive_c" type="Events:tliMReceive_c"/>

<!-- synchronous communication -->
<xsd:element name="tliPrCall_m" type="Events:tliPrCall_m"/>
<xsd:element name="tliPrCall_c" type="Events:tliPrCall_c"/>
<xsd:element name="tliPrCall_m_BC" type="Events:tliPrCall_m_BC"/>
<xsd:element name="tliPrCall_c_BC" type="Events:tliPrCall_c_BC"/>
<xsd:element name="tliPrCall_m_MC" type="Events:tliPrCall_m_MC"/>
<xsd:element name="tliPrCall_c_MC" type="Events:tliPrCall_c_MC"/>

<xsd:element name="tliPrGetCallDetected_m" type="Events:tliPrGetCallDetected_m"/>
<xsd:element name="tliPrGetCallDetected_c" type="Events:tliPrGetCallDetected_c"/>
<xsd:element name="tliPrGetCallMismatch_m" type="Events:tliPrGetCallMismatch_m"/>
<xsd:element name="tliPrGetCallMismatch_c" type="Events:tliPrGetCallMismatch_c"/>
<xsd:element name="tliPrGetCall_m"      type="Events:tliPrGetCall_m"/>
<xsd:element name="tliPrGetCall_c"      type="Events:tliPrGetCall_c"/>

<xsd:element name="tliPrReply_m"    type="Events:tliPrReply_m"/>
<xsd:element name="tliPrReply_c"    type="Events:tliPrReply_c"/>
<xsd:element name="tliPrReply_m_BC" type="Events:tliPrReply_m_BC"/>
<xsd:element name="tliPrReply_c_BC" type="Events:tliPrReply_c_BC"/>
<xsd:element name="tliPrReply_m_MC" type="Events:tliPrReply_m_MC"/>
<xsd:element name="tliPrReply_c_MC" type="Events:tliPrReply_c_MC"/>

<xsd:element name="tliPrGetReplyDetected_m" type="Events:tliPrGetReplyDetected_m"/>
<xsd:element name="tliPrGetReplyDetected_c" type="Events:tliPrGetReplyDetected_c"/>
<xsd:element name="tliPrGetReplyMismatch_m" type="Events:tliPrGetReplyMismatch_m"/>
<xsd:element name="tliPrGetReplyMismatch_c" type="Events:tliPrGetReplyMismatch_c"/>
<xsd:element name="tliPrGetReply_m"     type="Events:tliPrGetReply_m"/>
<xsd:element name="tliPrGetReply_c"     type="Events:tliPrGetReply_c"/>

<xsd:element name="tliPrRaise_m"    type="Events:tliPrRaise_m"/>
<xsd:element name="tliPrRaise_c"    type="Events:tliPrRaise_c"/>
<xsd:element name="tliPrRaise_m_BC" type="Events:tliPrRaise_m_BC"/>
<xsd:element name="tliPrRaise_c_BC" type="Events:tliPrRaise_c_BC"/>
<xsd:element name="tliPrRaise_m_MC" type="Events:tliPrRaise_m_MC"/>
<xsd:element name="tliPrRaise_c_MC" type="Events:tliPrRaise_c_MC"/>

<xsd:element name="tliPrCatchDetected_m"     type="Events:tliPrCatchDetected_m"/>
<xsd:element name="tliPrCatchDetected_c"     type="Events:tliPrCatchDetected_c"/>
<xsd:element name="tliPrCatchMismatch_m"     type="Events:tliPrCatchMismatch_m"/>
<xsd:element name="tliPrCatchMismatch_c"     type="Events:tliPrCatchMismatch_c"/>
<xsd:element name="tliPrCatch_m"     type="Events:tliPrCatch_m"/>
<xsd:element name="tliPrCatch_c"     type="Events:tliPrCatch_c"/>

<xsd:element name=" tliPrCatchTimeoutDetected "
                           type="Events:tliPrCatchTimeoutDetected "/>
<xsd:element name="tliPrCatchTimeout"    type="Events:tliPrCatchTimeout"/>

<!-- components -->
<xsd:element name="tliCCreate" type="Events:tliCCreate"/>
<xsd:element name="tliCStart" type="Events:tliCStart"/>
<xsd:element name="tliCRunning" type="Events:tliCRunning"/>
<xsd:element name="tliCAlive" type="Events:tliCAlive"/>
<xsd:element name="tliCStop" type="Events:tliCStop"/>
<xsd:element name="tliCKill" type="Events:tliCKill"/>
<xsd:element name="tliCDoneMismatch" type="Events:tliCDoneMismatch"/>
<xsd:element name="tliCDone" type="Events:tliCDone"/>
<xsd:element name="tliCKilledMismatch" type="Events:tliCKilledMismatch"/>
<xsd:element name="tliCKilled" type="Events:tliCKilled"/>
<xsd:element name="tliCTerminated" type="Events:tliCTerminated"/>

<!-- ports -->
<xsd:element name="tliPConnect" type="Events:tliPConnect"/>
<xsd:element name="tliPDisconnect" type="Events:tliPDisconnect"/>
<xsd:element name="tliPMap" type="Events:tliPMap"/>
<xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>
<xsd:element name="tliPClear" type="Events:tliPClear"/>
<xsd:element name="tliPStart" type="Events:tliPStart"/>
<xsd:element name="tliPStop" type="Events:tliPStop"/>
<xsd:element name="tliPHalt" type="Events:tliPHalt"/>

<!-- codec -->
<xsd:element name="tliDecode" type="Events:tliDecode"/>
<xsd:element name="tliEncode" type="Events:tliEncode"/>

<!-- timers -->
<xsd:element name="tliTTimeoutDetected" type="Events:tliTTimeoutDetected"/>
```

```
            <xsd:element name="tliTTimeoutMismatch" type="Events:tliTTimeoutMismatch"/>
            <xsd:element name="tliTTimeout" type="Events:tliTTimeout"/>
            <xsd:element name="tliTStart" type="Events:tliTStart"/>
            <xsd:element name="tliTStop" type="Events:tliTStop"/>
            <xsd:element name="tliTRead" type="Events:tliTRead"/>
            <xsd:element name="tliTRunning" type="Events:tliTRunning"/>

            <!-- scopes -->
            <xsd:element name="tliSEnter" type="Events:tliSEnter"/>
            <xsd:element name="tliSLeave" type="Events:tliSLeave"/>

            <!-- statements -->
            <xsd:element name="tliVar" type="Events:tliVar"/>
            <xsd:element name="tliModulePar" type="Events:tliModulePar"/>
            <xsd:element name="tliGetVerdict" type="Events:tliGetVerdict"/>
            <xsd:element name="tliSetVerdict" type="Events:tliSetVerdict"/>
            <xsd:element name="tliLog" type="Events:tliLog"/>

            <!-- alt  -->
            <xsd:element name="tliAEnter" type="Events:tliAEnter"/>
            <xsd:element name="tliALeave" type="Events:tliALeave"/>
            <xsd:element name="tliADefaults" type="Events:tliADefaults"/>
            <xsd:element name="tliAActivate" type="Events:tliAActivate"/>
            <xsd:element name="tliADeactivate" type="Events:tliADeactivate"/>
            <xsd:element name="tliANomatch" type="Events:tliANomatch"/>
            <xsd:element name="tliARepeat" type="Events:tliARepeat"/>
            <xsd:element name="tliAWait" type="Events:tliAWait"/>

            <!-- action  -->
            <xsd:element name="tliAction" type="Events:tliAction"/>

            <!-- match  -->
            <xsd:element name="tliMatch" type="Events:tliMatch"/>
            <xsd:element name="tliMatchMismatch" type="Events:tliMatchMismatch"/>

            <!-- info  -->
            <xsd:element name="tliInfo" type="Events:tliInfo"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:schema>
```

Bibliography

[b-CGI/NPL038]   INTOOL CGI/NPL038 (V2.2) (1996), *Generic Compiler/Interpreter interface; GCI Interface Specification* Infrastructural Tools, December.

[b-ITU-T X.292]   ITU-T Recommendation X.292 (2002), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – The Tree and Tabular Combined Notation (TTCN).*

[ISO/IEC 10646]   ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS).*
<http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue-detail.htm?csnumber=39921>

[b-OMG CORBA]   OMG CORBA v2.2 (2004), *The Common Object Request Broker: Architecture and Specification.*
<http://www.omg.org/technology/documents/formal/corba_2.htm>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	**Languages and general software aspects for telecommunication systems**