

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.145

(03/2006)

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)

ITU-T Recommendation Z.145

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
Extended Object Definition Language (eODL)	Z.130–Z.139
Testing and Test Control Notation (TTCN)	Z.140–Z.149
User Requirements Notation (URN)	Z.150–Z.159
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	
Distributed processing environment	Z.600–Z.609

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

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

Summary

This Recommendation specifies the control interfaces for TTCN-3 (*Testing and Test Control Notation 3*) test system implementations. The TTCN-3 Control Interface provides the recommended 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 ITU-T Rec. Z.140. The interface definition in this Recommendation uses the CORBA Interface Definition Language (IDL) to specify the TCI completely. Clauses 8 and 9 specify language mappings of the abstract specification to the target languages Java and ANSI-C. A summary of the IDL-based interface specification is provided in Annex A.

Source

ITU-T Recommendation Z.145 was approved on 16 March 2006 by ITU-T Study Group 17 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. 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 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.

© ITU 2006

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

	<i>Page</i>
1 Scope	1
2 References	1
3 Definitions and abbreviations	2
3.1 Definitions	2
3.2 Abbreviations	3
4 Introduction	3
5 Compliance	4
6 General structure of a TTCN-3 test system	4
6.1 Entities in a TTCN-3 test system	4
6.2 Execution requirements for a TTCN-3 test system	7
7 TTCN-3 control interface and operations	7
7.1 Overview of the TCI	7
7.2 TCI data	9
7.3 TCI operations	18
8 Java language mapping	83
8.1 Introduction	83
8.2 Names and scopes	83
8.3 Constants	97
8.4 Mapping of interfaces	98
8.5 Optional parameters	105
8.6 TCI initialization	105
8.7 Error handling	105
9 ANSI-C language mapping	105
9.1 Introduction	105
9.2 Value interfaces	106
9.3 Logging interface	109
9.4 Operation interfaces	109
9.5 Data	123
9.6 Miscellaneous	125
10 W3C XML mapping	125
10.1 Introduction	125
10.2 Scopes	125
10.3 Type mapping	126
10.4 Mapping of the operations on the logging interface	144
11 Use scenarios	166
11.1 Initialization, collecting information, logging	166
11.2 Execution of test cases and control	169
11.3 Component handling	172
11.4 Termination of test cases and control	179
11.5 Communication	184
Annex A – IDL specification of TCI	189
Annex B – XML mapping for TCI TL provided	204
B.1 TCI-TL XML Schema for Simple Types	204
B.2 TCI-TL XML Schema for Types	204
B.3 TCI-TL XML Schema for Values	206
B.4 TCI-TL XML Schema for Templates	211
B.5 TCI-TL XML Schema for Events	218
B.6 TCI-TL XML Schema for a Log	236
BIBLIOGRAPHY	238

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 standard (see references below). 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.

- [1] ITU-T Recommendation Z.144 (2006), *Testing and Test Control Notation version 3 (TTCN-3): Runtime interface (TRI)*.
- [2] ITU-T Recommendation Z.140 (2006), *Testing and Test Control Notation version 3 (TTCN-3): Core language*.
- [3] ITU-T Recommendation Z.143 (2006), *Testing and Test Control Notation version 3 (TTCN-3) Operational semantics*.
- [4] 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*.
- [5] W3C Recommendation (2004), *XML Schema Part 0: Primer*.
NOTE – See at <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>.
- [6] W3C Recommendation (2004), *XML Schema Part 1: Structures*.
NOTE – See at <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.
- [7] W3C Recommendation (2004), *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

For the purpose of this Recommendation, the terms and definitions given in ITU-T Rec. X.290 [4] and the following apply:

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 incl. 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 Rec. X.290 [4].

3.1.8 Implementation eXtra Information for Testing (IXIT): Refer to ITU-T Rec. X.290 [4].

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 Rec. X.290 [4].

3.1.11 System Under Test (SUT): Refer to ITU-T Rec. X.290 [4].

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 Rec. X.290 [4].

3.1.14 test case: Refer to ITU-T Rec. X.290 [4].

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 Rec. X.290 [4].

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

For the purposes of this Recommendation, the following abbreviations apply:

ATS	Abstract Test Suite
CD	Coding/Decoding
CH	Component Handling
ETS	Executable Test Suite
IDL	Interface Definition Language
IXIT	Implementation eXtra Information for Testing
MSC	Message Sequence Chart
MTC	Main Test Component
OMG	Object Management Group
PA	Platform Adaptor
PTC	Parallel Test Component
SA	SUT Adaptor
SUT	System Under Test
TC	Test Control
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); and
- 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 Rec. Z.144 [1]).

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 Rec. Z.143 [3]. 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.

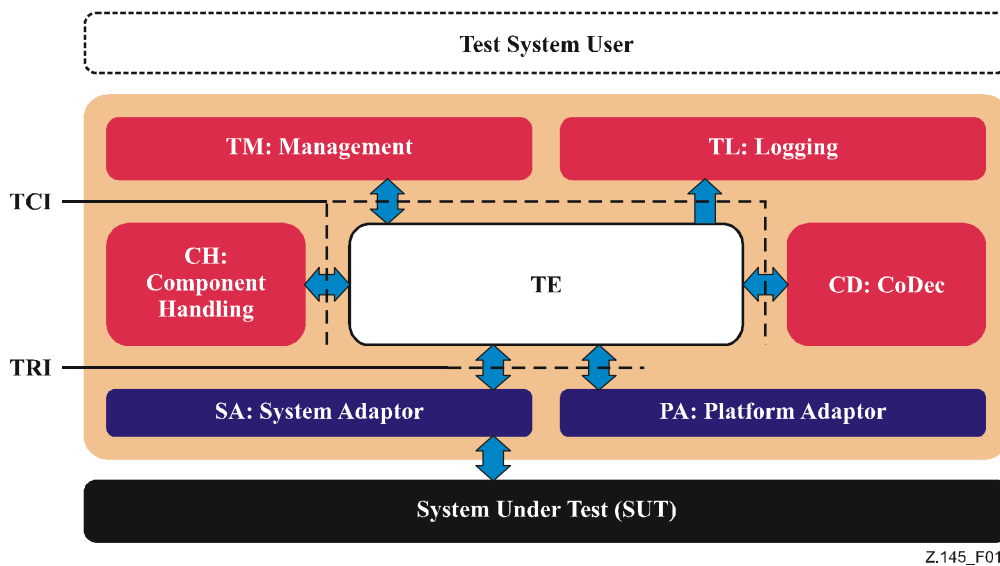


Figure 1/Z.145 – General structure of a TTCN-3 test system

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 Rec. Z.140 [2]) 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 Rec. Z.144 [1]. 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.

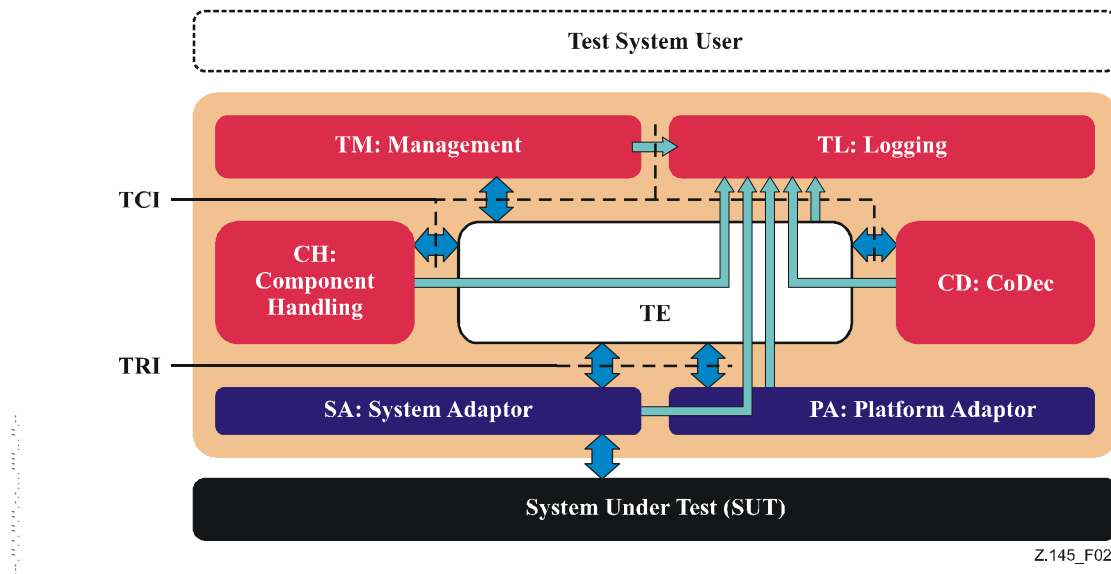


Figure 2/Z.145 – Detailed View on TL

6.1.1 Test Management and Control (TMC)

The TMC entity includes functionality related to management of:

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

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 EXIT 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 encoding and decoding of TTCN-3 values into bitstrings suitable to be sent to the System Under Test. 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.

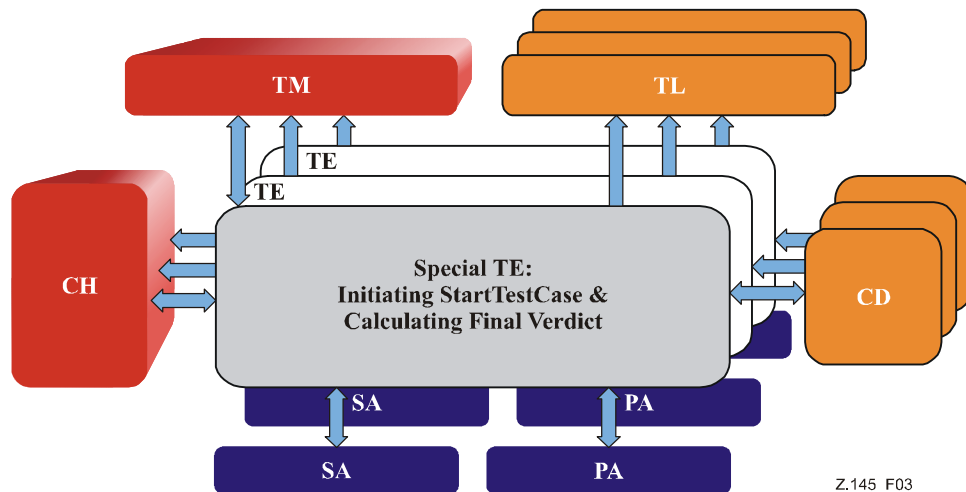


Figure 3/Z.145 – General structure of a distributed TTCN-3 test system

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 – A test system shall execute at most one test case at a given point in time. That is, 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 Rec. Z.144 [1]. The terminology for TE defined in ITU-T Rec. Z.144 [1] 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 [1]. The terminology for SA defined in ITU-T Rec. Z.144 [1] 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 Rec. Z.144 [1]. The terminology for PA defined in ITU-T Rec. Z.144 [1] 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 this 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 operation 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 Rec. Z.144 [1].

Table 1/Z.145 – 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
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	tciumapReq	TCI-CH Provided
	tciumap	TCI-CH Required
running	tcitestComponentRunningReq	TCI-CH Provided
	tcitestComponentRunning	TCI-CH Required

Table 1/Z.145 – Correlation between TTCN-3 and TCI operation invocations

TTCN-3 operation name	TCI operation name	TCI interface name
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
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

<code>TciModuleIdType</code>	A value of <code>TciModuleIdType</code> is the name of a TTCN-3 module as specified in the TTCN-3 ATS. This abstract type is used for module handling.
<code>TciModuleParameterIdType</code>	A value of <code>TciModuleParameterIdType</code> 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.
<code>TciTestCaseIdType</code>	A value of <code>TciTestCaseIdType</code> is the qualified name of a TTCN-3 testcase as specified in the TTCN-3 ATS. This abstract type is used for testcase handling.
<code>TciModuleIdListType</code>	A value of type <code>TciModuleIdListType</code> is a list of <code>TciModuleIdType</code> . This abstract type is used when retrieving the list of modules which are imported by a TTCN-3 module.
<code>TciModuleParameterType</code>	A value of type <code>TciModuleParameterType</code> is a structure of <code>TciModuleParameterIdType</code> and <code>value</code> . This abstract type is used to represent the parameter name and the default value of a module parameter.

<code>TciModuleParameterListType</code>	A value of type <code>TciModuleParameterListType</code> is a list of <code>TciModuleParameterType</code> . This abstract type is used when retrieving the module parameters of a TTCN-3 module.
<code>TciParameterType</code>	A value of type <code>TciParameterType</code> includes a TTCN-3 value and a value of <code>TciParameterPassingModeType</code> to represent the parameter passing mode specified for the parameter in the TTCN-3 ATS.
<code>TciParameterPassingModeType</code>	A value of type <code>TciParameterPassingModeType</code> 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.
<code>TciParameterListType</code>	A value of type <code>TciParameterListType</code> is a list of <code>TciParameterType</code> . This abstract type is used when starting a test case or when the termination of a test case is indicated.
<code>TciParameterTypeType</code>	A value of type <code>TciParameterTypeType</code> is a structure of <code>Type</code> and <code>TciParameterPassingModeType</code> . This abstract type is used to represent the type and the parameter passing mode of a test case parameter.
<code>TciParameterTypeListType</code>	A value of type <code>TciParameterTypeListType</code> is a list of <code>TciParameterTypeType</code> . This abstract type is used to represent the list of parameters of a test case.
<code>TciTestComponentKindType</code>	A value of type <code>TciTestComponentKindType</code> is a literal of the set of kinds of TTCN-3 test components, i.e., MTC, PTC, PTC_ALIVE, and CONTROL. This abstract type is used for component handling.
<code>TciTypeClassType</code>	A value of type <code>TciTypeClassType</code> 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 Rec. Z.144 [1]: `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`:

<code>TciModuleIdType getDefiningModule()</code>	Returns the module identifier of the module in which type is defined. Returns the distinct value <code>null</code> if type is a TTCN-3 base type, e.g., boolean, integer, etc.
<code>TString getName()</code>	Returns the name of the type as defined in the TTCN-3 module.
<code>TciTypeClassType getTypeClass()</code>	Returns the type class of the respective type. A value of <code>TciTypeClassType</code> can have one of the following constants: <code>ADDRESS</code> , <code>ANYTYPE</code> , <code>BITSTRING</code> , <code>BOOLEAN</code> , <code>CHARSTRING</code> , <code>COMPONENT</code> , <code>ENUMERATED</code> , <code>FLOAT</code> , <code>HEXSTRING</code> , <code>INTEGER</code> , <code>OBJID</code> , <code>OCTETSTRING</code> , <code>RECORD</code> , <code>RECORD_OF</code> , <code>SET</code> , <code>SET_OF</code> , <code>UNION</code> , <code>UNIVERSAL_CHARSTRING</code> , <code>VERDICT</code> .
<code>Value newInstance()</code>	Returns a freshly created value of the given type. This initial value of the created value is undefined.
<code>TString getTypeEncoding()</code>	Returns the type encoding attribute as defined in the TTCN-3 module.
<code>TString getTypeEncodingVariant()</code>	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 <code>null</code> is returned.
<code>TStringseq getTypeExtension()</code>	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).

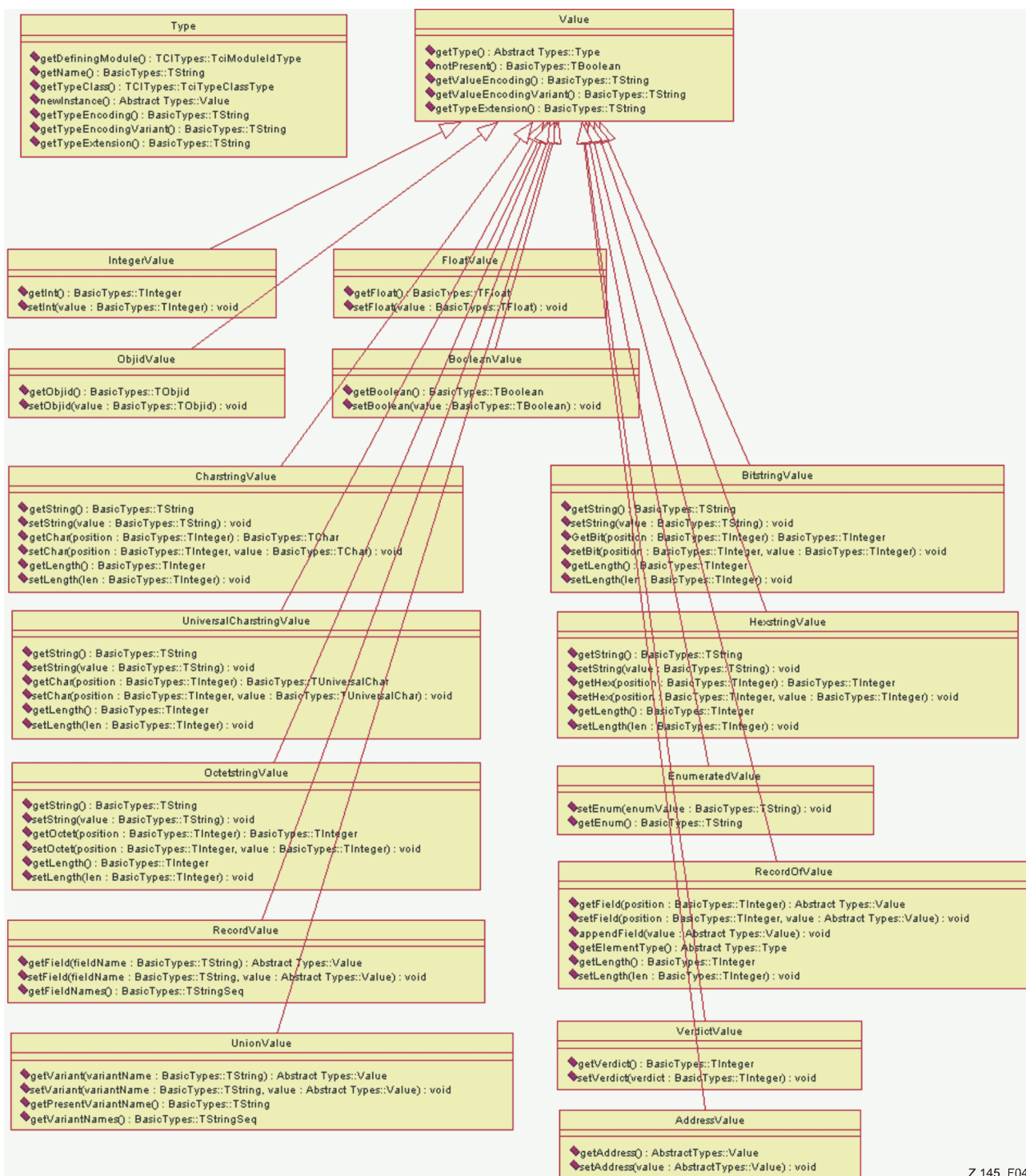


Figure 4/Z.145 – Hierarchy of abstract values

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 <code>getType()</code>	Returns the type of the specified value.
TBoolean <code>notPresent()</code>	Returns <code>true</code> if the specified value is omit, <code>false</code> otherwise.

<code>TString getValueEncoding()</code>	Returns the value encoding attribute as defined in TTCN-3, if any. If no encoding attribute is defined, the distinct value <code>null</code> is returned.
<code>TString getValueEncodingVariant()</code>	Returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute is defined, the distinct value <code>null</code> 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`:

<code>TInteger getInt()</code>	Returns the integer value of this TTCN-3 integer.
<code>void setInt(in TInteger value)</code>	Sets this <code>IntegerValue</code> 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`:

<code>TFloat getFloat()</code>	Returns the float value of this TTCN-3 float.
<code>void setFloat(in TFloat value)</code>	Sets this <code>FloatValue</code> 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 `BooleanValue`:

<code>TBoolean getBoolean()</code>	Returns the boolean value of the TTCN-3 boolean.
<code>void setBoolean(in TBoolean value)</code>	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`:

<code>TObjid getObjid()</code>	Returns the object id value of the TTCN-3 <code>objid</code> .
<code>void setObjid(in TObjid value)</code>	Sets this <code>ObjidValue</code> 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`:

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

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`:

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

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

<code>TString getString()</code>	Returns the textual representation of this <code>OctetstringValue</code> , as defined in TTCN-3. E.g., the textual representation of <code>0xCAFFEE</code> is <code>"CAFFEE"O</code> . The textual representation of the empty TTCN-3 <code>octetstring</code> is <code>"O</code> , while its length is zero.
<code>void setString(in TString value)</code>	Sets the value of this <code>OctetstringValue</code> according to the textual representation as defined by <code>value</code> . E.g., The value of this <code>OctetstringValue</code> is <code>0xCAFFEE</code> if the textual representation in <code>value</code> is <code>"CAFFEE"O</code> .
<code>TChar getOctet(in TInteger position)</code>	Returns the value (0..255) at position of this TTCN-3 <code>octetstring</code> . Position 0 denotes the first octet of the TTCN-3 <code>octetstring</code> . Valid values for position are from 0 to <code>length - 1</code> .
<code>void setOctet(in TInteger position, in TInteger value)</code>	Sets the octet at position to value (0..255). Position 0 denotes the first octet in the <code>octetstring</code> . Valid values for position are from 0 to <code>length - 1</code> .
<code>TInteger getLength()</code>	Returns the length of this <code>OctetstringValue</code> in octets, zero if the value of this <code>OctetstringValue</code> is <code>omit</code> .
<code>void setLength(in TInteger len)</code>	<code>setLength</code> first resets this <code>OctetstringValue</code> to its initial value and afterwards sets the length of this <code>OctetstringValue</code> in octets to <code>len</code> .

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`:

<code>TString getString()</code>	Returns the textual representation of this <code>HexstringValue</code> , as defined in TTCN-3. E.g., the textual representation of <code>0xAFFEE</code> is <code>"AFFEE"H</code> . The textual representation of the empty TTCN-3 <code>hexstring</code> is <code>"H</code> , while its length is zero.
<code>void setString(in TString value)</code>	Sets the value of this <code>HexstringValue</code> according to the textual representation as defined by <code>value</code> . E.g., The value of this <code>HexstringValue</code> is <code>0xAFFEE</code> if the textual representation in <code>value</code> is <code>"AFFEE"H</code> .
<code>TChar getHex(in TInteger position)</code>	Returns the value (0..15) at position of this TTCN-3 <code>hexstring</code> . Position 0 denotes the first hex digits of the TTCN-3 <code>hexstring</code> . Valid values for position are from 0 to <code>length - 1</code> .
<code>void setHex(in TInteger position, in TInteger value)</code>	Sets the hex digit at position to value (0..15). Position 0 denotes the first octet in the <code>hexstring</code> . Valid values for position are from 0 to <code>length - 1</code> .
<code>TInteger getLength()</code>	Returns the length of this <code>HexstringValue</code> in octets, zero if the value of this <code>HexstringValue</code> is <code>omit</code> .
<code>void setLength(in TInteger len)</code>	<code>setLength</code> first resets this <code>HexstringValue</code> to its initial value and afterwards sets the length of this <code>HexstringValue</code> in hex digits to <code>len</code> .

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`:

<code>Value getField(in TString fieldName)</code>	Returns the value of the field named <code>fieldName</code> . The return value is the common abstract base type <code>Value</code> , 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 reflected in the record.

```
TStringSeq getFieldNames()
```

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

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 reflected 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 reflected 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 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 are 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:

<code>boolean isAny()</code>	Returns <code>true</code> if the template is an any template.
<code>boolean isAll()</code>	Returns <code>true</code> if the template is an all template.
<code>TString getTemplateDef()</code>	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:

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

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).

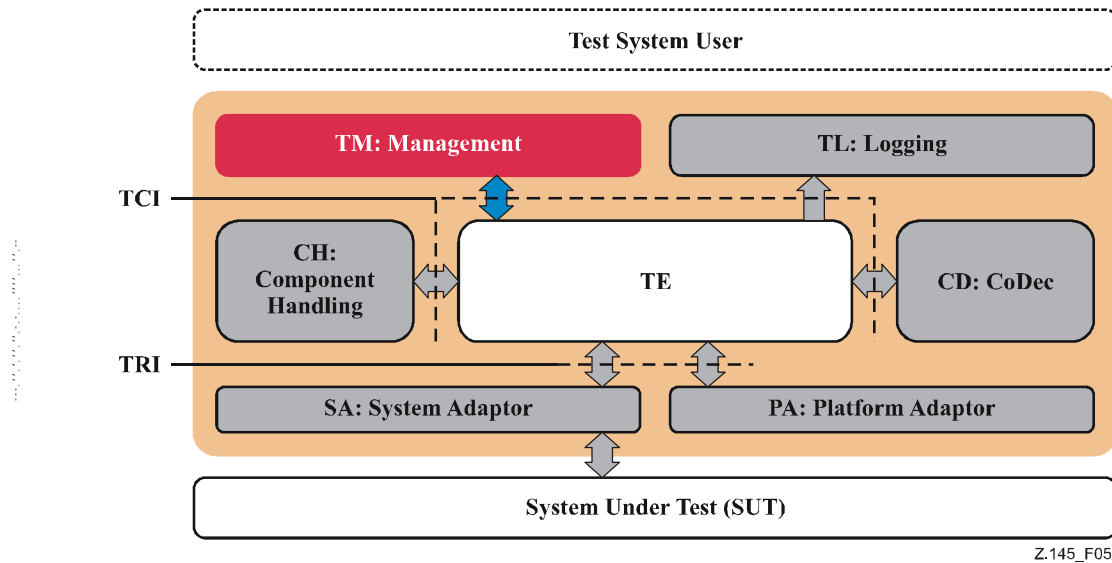


Figure 5/Z.145 – The TCI-TM interface

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

7.3.1.2.2 tciTestCaseTerminated

Signature	<code>void tciTestCaseTerminated(in VerdictValue verdict, in TciParameterListType parameterList)</code>	
In Parameters	<code>verdict</code>	The final verdict of the test case.
	<code>parameterList</code>	A list of values that are part of the test case signature. The parameters in <code>parameterList</code> are ordered as they appear in the TTCN-3 test case declaration.
Return Value	<code>void</code>	
Constraint	Shall only be called after either the control part of the module or a test cases has been started using the <i>required</i> operations <code>tciStartControl</code> or <code>tciStartTestCase</code> .	
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 <code>verdict</code> . On the invocation of a <code>tciTestCaseTerminated</code> operation, all <i>out</i> and <i>inout</i> test case parameters contain values. All in test case parameters contain the distinct value of <code>null</code> 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	<code>void tciControlTerminated()</code>	
Return Value	<code>void</code>	
Constraint	Shall only be called when the module execution has been started using the <code>tciStartControl</code> 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	<code>Value tciGetModulePar(in TciModuleParameterIdType parameterId)</code>	
In Parameters	<code>parameterId</code>	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 <code>tciStartTestCase</code> and <code>tciTestCaseTerminated</code> pair if a test case has been started explicitly or between a <code>tciStartControl</code> and <code>tciControlTerminated</code> pair if the control part of a module has been started.	
Effect	The management provides to the TE a <code>Value</code> for the indicated <code>parameterId</code> . Every call of <code>tciGetModulePar()</code> 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 <code>null</code> 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).

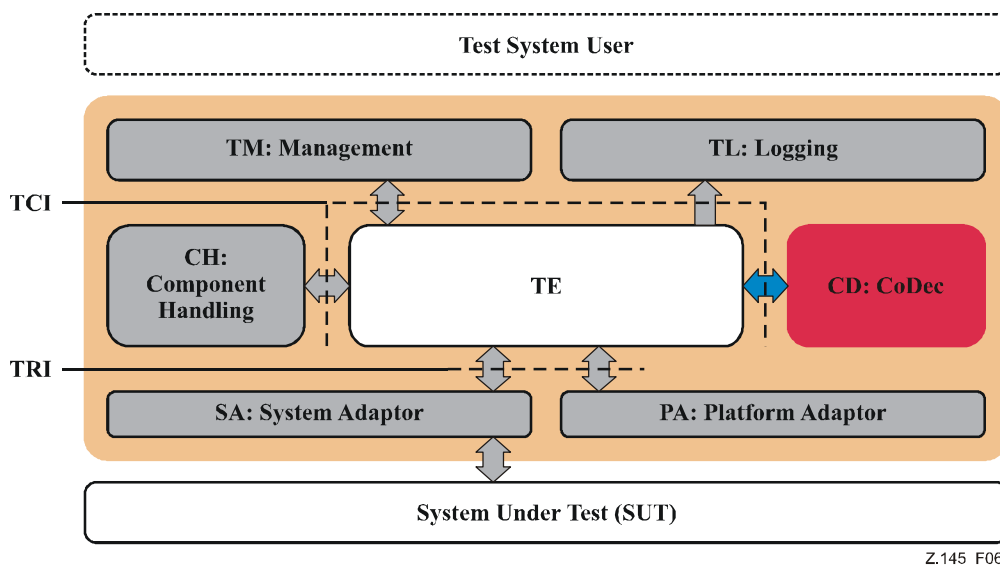


Figure 6/Z.145 – The TCI-CD interface

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 charstringtype.

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 a encoded value, or might for performance considerations postpone the decoding until the actual access of the encoded value.	
Effect	This operation decodes 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).

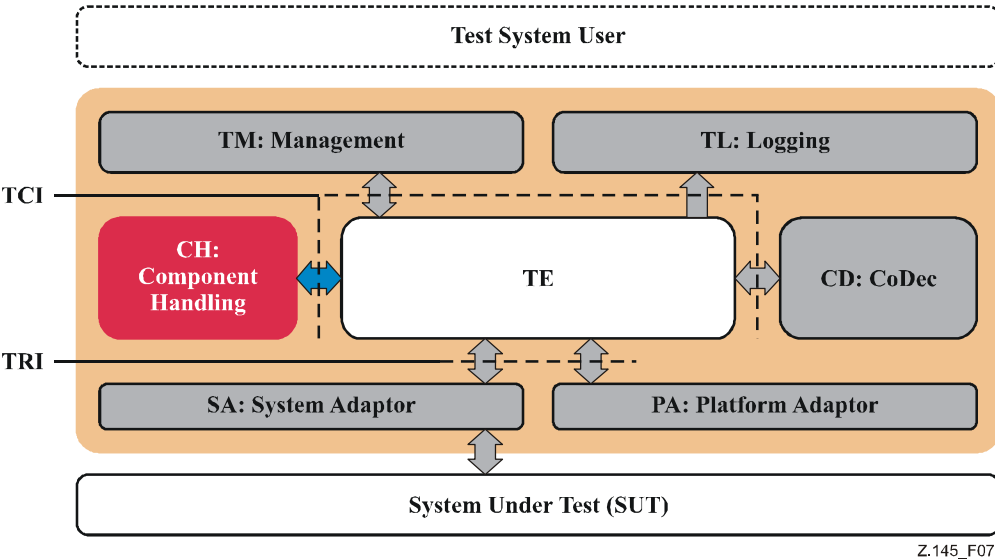


Figure 7/Z.145 – The TCI-CH interface

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" always refers 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 `tciParamStartTestCase()` or `tciParamStartControl()` 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 tciParamEnqueueMsgConnected

Signature	void tciParamEnqueueMsgConnected(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 <i>provided</i> tciParamSendConnected 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 tciParamEnqueueCallConnected

Signature	void tciParamEnqueueCallConnected(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 <i>provided</i> tciParamCallConnected has been called. All <i>in</i> and <i>inout</i> procedure parameters contain values. All <i>out</i> procedure parameters shall contain the distinct value of <code>null</code> 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 <i>provided</i> tciReplyConnected has been called. All <i>out</i> and <i>inout</i> procedure parameters and the return value contain values. All <i>in</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> tciStartTestComponentReq has been called. Since only <i>in</i> parameters are allowed for functions being started (ITU-T Rec. Z.140 [2]), parameterList contains only <i>in</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>in</i> parameters (ITU-T Rec. Z.140 [2]), 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>provided</i> 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 <i>in</i> and <i>inout</i> procedure parameters contain values. All <i>out</i> 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 <i>in</i> and <i>inout</i> procedure parameters contain values. All <i>out</i> 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 <i>in</i> and <i>inout</i> procedure parameters contain values. All <i>out</i> 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 <i>out</i> and <i>inout</i> procedure parameters and the return value contain values. All <i>in</i> 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 <i>out</i> and <i>inout</i> procedure parameters and the return value contain values. All <i>in</i> procedure parameters shall contain the distinct value of <code>null</code> since they are only of relevance to the procedure call but not in the reply to the call. The <code>parameterList</code> 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 <code>null</code> 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 <code>signature</code> and all components connected to <code>sender</code> . 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 <i>out</i> and <i>inout</i> procedure parameters and the return value contain values. All <i>in</i> procedure parameters shall contain the distinct value of <code>null</code> since they are only of relevance to the procedure call but not in the reply to the call. The <code>parameterList</code> 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 <code>null</code> 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 <code>signature</code> and one of the component identifier in <code>receivers</code> . CH transmits the reply to the remote TEs on which <code>receivers</code> 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 <i>signature</i> and component identifier <i>receiver</i> . CH transmits the exception to the remote TE on which <i>receiver</i> 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 <i>signature</i> and all components connected to <i>sender</i> . 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 receiver, 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 <i>signature</i> and one of the component identifier <i>receivers</i> . CH transmits the exception to all remote TEs on which <i>receivers</i> 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 Rec. Z.140 [2]), 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 <i>in</i> parameters (ITU-T Rec. Z.140 [2]), 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).

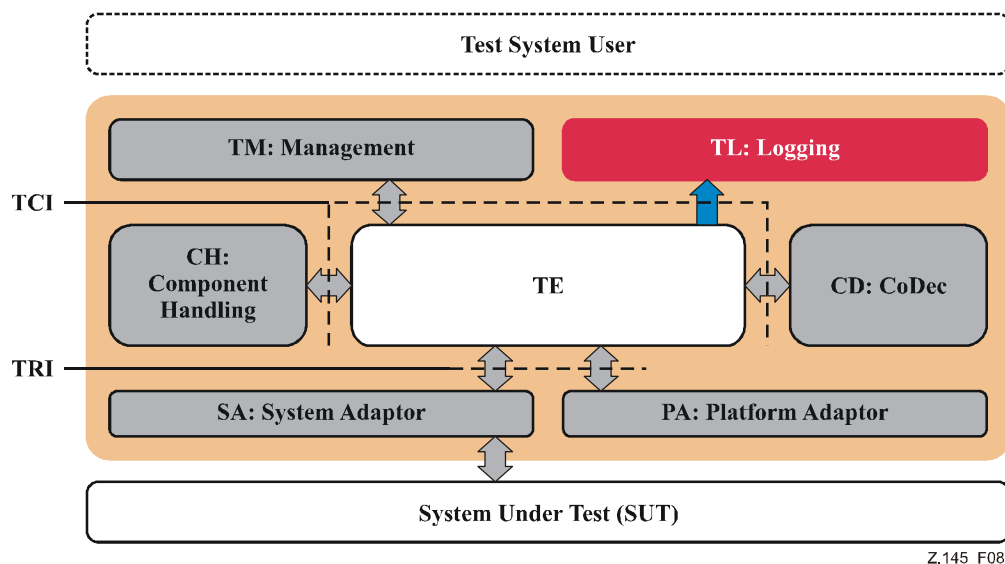


Figure 8/Z.145 – The TCI-TL interface

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 pars, 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.
	pars	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 TriParameterListType pars, 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.
	pars	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 TriParameterListType pars, 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.
	pars	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 TriParameterListType pars, in VerdictValue outcome)	
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.
	pars	The list of parameters required by the testcase.
	outcome	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 port, in Value msgValue, in TriAddressType address, in TriStatusType 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.
	port	The port via 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 port, in Value msgValue, in TriStatusType 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.
	port	The port via 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 port, in Value msgValue, in TriAddressListType addresses, in TriStatusType 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.
	port	The port via 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.	