

**Methods for Testing and Specification (MTS);
The Testing and Test Control Notation version 3;
Part 6: TTCN-3 Control Interface (TCI)**



Reference

RES/MTS-00107-6 T3 ed411 TCI

Keywords

control, interface, MTS, TCI, testing, TTCN

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

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

© European Telecommunications Standards Institute 2009.
All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™**, **TIPHON™**, the TIPHON logo and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.

3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

LTE™ is a Trade Mark of ETSI currently being registered

for the benefit of its Members and of the 3GPP Organizational Partners.

GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights	13
Foreword.....	13
1 Scope	14
2 References	14
2.1 Normative references	14
2.2 Informative references.....	15
3 Definitions and abbreviations.....	15
3.1 Definitions.....	15
3.2 Abbreviations	16
4 Introduction	17
5 Compliance.....	17
6 General structure of a TTCN-3 test system.....	17
6.1 Entities in a TTCN-3 test system.....	18
6.1.1 Test Management and Control (TMC).....	19
6.1.1.1 Test Management (TM)	19
6.1.1.2 Coding and Decoding (CD)	19
6.1.1.3 Component Handling (CH).....	19
6.1.1.4 Test Logging (TL).....	21
6.1.2 TTCN-3 Executable (TE)	21
6.1.3 SUT Adaptor (SA).....	21
6.1.4 Platform Adaptor (PA).....	21
6.2 Execution requirements for a TTCN-3 test system	21
7 TTCN-3 control interface and operations.....	21
7.1 Overview of the TCI.....	21
7.1.1 Correlation between TTCN-3 and TCI operation invocations	22
7.2 TCI data.....	23
7.2.1 General abstract data types	23
7.2.1.1 Management.....	23
7.2.1.2 Communication	24
7.2.2 Abstract TTCN-3 data types and values	24
7.2.2.1 Abstract TTCN-3 data types	25
7.2.2.2 Abstract TTCN-3 values	26
7.2.2.2.1 The abstract data type Value	26
7.2.2.2.2 The abstract data type IntegerValue	27
7.2.2.2.3 The abstract data type FloatValue	27
7.2.2.2.4 The abstract data type BooleanValue	27
7.2.2.2.5 The abstract data type CharstringValue	27
7.2.2.2.6 The abstract data type UniversalCharstringValue.....	28
7.2.2.2.7 The abstract data type BitstringValue.....	28
7.2.2.2.8 The abstract data type OctetstringValue.....	29
7.2.2.2.9 The abstract data type HexstringValue.....	29
7.2.2.2.10 The abstract data type RecordValue.....	30
7.2.2.2.11 The abstract data type RecordOfValue	30
7.2.2.2.12 The abstract data type UnionValue	31
7.2.2.2.13 The abstract data type EnumeratedValue	31
7.2.2.2.14 The abstract data type VerdictValue	32
7.2.2.2.15 The abstract data type AddressValue	32
7.2.3 Abstract logging types	32
7.2.3.1 The abstract data type TciValueTemplate.....	32
7.2.3.2 The abstract data type TciNonValueTemplate	32
7.2.3.3 The Value List and Mismatch Types	32

7.2.3.4	The Status Types	33
7.3	TCI operations	33
7.3.1	The TCI-TM interface	34
7.3.1.1	TCI-TM required.....	35
7.3.1.1.1	tciRootModule.....	35
7.3.1.1.2	tciGetImportedModules.....	35
7.3.1.1.3	tciGetModuleParameters	35
7.3.1.1.4	tciGetTestCases	35
7.3.1.1.5	tciGetTestCaseParameters	36
7.3.1.1.6	tciGetTestCaseTSI.....	36
7.3.1.1.7	tciStartTestCase	36
7.3.1.1.8	tciStopTestCase	36
7.3.1.1.9	tciStartControl	37
7.3.1.1.10	tciStopControl	37
7.3.1.2	TCI-TM provided.....	37
7.3.1.2.1	tciTestCaseStarted	37
7.3.1.2.2	tciTestCaseTerminated	37
7.3.1.2.3	tciControlTerminated	38
7.3.1.2.4	tciGetModulePar.....	38
7.3.1.2.5	tciLog	38
7.3.1.2.6	tciError.....	38
7.3.2	The TCI-CD interface	38
7.3.2.1	TCI-CD required	39
7.3.2.1.1	getTypeForName	39
7.3.2.1.2	getInteger	40
7.3.2.1.3	getFloat	40
7.3.2.1.4	getBoolean.....	40
7.3.2.1.5	Void	40
7.3.2.1.6	getCharstring	40
7.3.2.1.7	getUniversalCharstring.....	40
7.3.2.1.8	getHexstring	40
7.3.2.1.9	getBitstring	40
7.3.2.1.10	getOctetstring	40
7.3.2.1.11	getVerdict	41
7.3.2.1.12	tciErrorReq	41
7.3.2.2	TCI-CD provided	41
7.3.2.2.1	decode.....	41
7.3.2.2.2	encode.....	41
7.3.3	The TCI-CH interface	41
7.3.3.1	TCI-CH required	42
7.3.3.1.1	tciEnqueueMsgConnected	43
7.3.3.1.2	tciEnqueueCallConnected	43
7.3.3.1.3	tciEnqueueReplyConnected.....	43
7.3.3.1.4	tciEnqueueRaiseConnected	44
7.3.3.1.5	tciCreateTestComponent	44
7.3.3.1.6	tciStartTestComponent	44
7.3.3.1.7	tciStopTestComponent	44
7.3.3.1.8	tciConnect.....	45
7.3.3.1.9	tciDisconnect	45
7.3.3.1.10	tciMap.....	45
7.3.3.1.11	tciUnmap	45
7.3.3.1.12	tciTestComponentTerminated	45
7.3.3.1.13	tciTestComponentRunning.....	46
7.3.3.1.14	tciTestComponentDone	46
7.3.3.1.15	tciGetMTC.....	46
7.3.3.1.16	tciExecuteTestCase.....	46
7.3.3.1.17	tciReset	47
7.3.3.1.18	tciKillTestComponent	47
7.3.3.1.19	tciTestComponentAlive.....	47
7.3.3.1.20	tciTestComponentKilled.....	47
7.3.3.2	TCI-CH provided	47
7.3.3.2.1	tciSendConnected	47

7.3.3.2.2	tciSendConnectedBC.....	48
7.3.3.2.3	tciSendConnectedMC.....	48
7.3.3.2.4	tciCallConnected.....	48
7.3.3.2.5	tciCallConnectedBC.....	49
7.3.3.2.6	tciCallConnectedMC.....	49
7.3.3.2.7	tciReplyConnected.....	50
7.3.3.2.8	tciReplyConnectedBC.....	50
7.3.3.2.9	tciReplyConnectedMC.....	51
7.3.3.2.10	tciRaiseConnected.....	51
7.3.3.2.11	tciRaiseConnectedBC.....	51
7.3.3.2.12	tciRaiseConnectedMC.....	52
7.3.3.2.13	tciCreateTestComponentReq.....	52
7.3.3.2.14	tciStartTestComponentReq.....	52
7.3.3.2.15	tciStopTestComponentReq.....	52
7.3.3.2.16	tciConnectReq.....	53
7.3.3.2.17	tciDisconnectReq.....	53
7.3.3.2.18	tciMapReq.....	53
7.3.3.2.19	tciUnmapReq.....	53
7.3.3.2.20	tciTestComponentTerminatedReq.....	54
7.3.3.2.21	tciTestComponentRunningReq.....	54
7.3.3.2.22	tciTestComponentDoneReq.....	54
7.3.3.2.23	tciGetMTCReq.....	54
7.3.3.2.24	tciExecuteTestCaseReq.....	55
7.3.3.2.25	tciResetReq.....	55
7.3.3.2.26	tciKillTestComponentReq.....	55
7.3.3.2.27	tciTestComponentAliveReq.....	55
7.3.3.2.28	tciTestComponentKilledReq.....	55
7.3.4	The TCI-TL interface.....	56
7.3.4.1	TCI-TL provided.....	56
7.3.4.1.1	tliTcExecute.....	56
7.3.4.1.2	tliTcStart.....	57
7.3.4.1.3	tliTcStop.....	57
7.3.4.1.4	tliTcStarted.....	57
7.3.4.1.5	tliTcTerminated.....	58
7.3.4.1.6	tliCtrlStart.....	58
7.3.4.1.7	tliCtrlStop.....	58
7.3.4.1.8	tliCtrlTerminated.....	59
7.3.4.1.9	tliMSend_m.....	59
7.3.4.1.10	tliMSend_m_BC.....	60
7.3.4.1.11	tliMSend_m_MC.....	60
7.3.4.1.12	tliMSend_c.....	61
7.3.4.1.13	tliMSend_c_BC.....	61
7.3.4.1.14	tliMSend_c_MC.....	61
7.3.4.1.15	tliMDetected_m.....	62
7.3.4.1.16	tliMDetected_c.....	62
7.3.4.1.17	tliMMismatch_m.....	63
7.3.4.1.18	tliMMismatch_c.....	63
7.3.4.1.19	tliMReceive_m.....	64
7.3.4.1.20	tliMReceive_c.....	64
7.3.4.1.21	tliPrCall_m.....	65
7.3.4.1.22	tliPrCall_m_BC.....	65
7.3.4.1.23	tliPrCall_m_MC.....	66
7.3.4.1.24	tliPrCall_c.....	66
7.3.4.1.25	tliPrCall_c_BC.....	67
7.3.4.1.26	tliPrCall_c_MC.....	67
7.3.4.1.27	tliPrGetCallDetected_m.....	68
7.3.4.1.28	tliPrGetCallDetected_c.....	68
7.3.4.1.29	tliPrGetCallMismatch_m.....	69
7.3.4.1.30	tliPrGetCallMismatch_c.....	69
7.3.4.1.31	tliPrGetCall_m.....	70
7.3.4.1.32	tliPrGetCall_c.....	70
7.3.4.1.33	tliPrReply_m.....	71

7.3.4.1.34	tliPrReply_m_BC	71
7.3.4.1.35	tliPrReply_m_MC	72
7.3.4.1.36	tliPrReply_c	72
7.3.4.1.37	tliPrReply_c_BC	73
7.3.4.1.38	tliPrReply_c_MC	73
7.3.4.1.39	tliPrGetReplyDetected_m	74
7.3.4.1.40	tliPrGetReplyDetected_c	74
7.3.4.1.41	tliPrGetReplyMismatch_m	75
7.3.4.1.42	tliPrGetReplyMismatch_c	75
7.3.4.1.43	tliPrGetReply_m	76
7.3.4.1.44	tliPrGetReply_c	76
7.3.4.1.45	tliPrRaise_m	77
7.3.4.1.46	tliPrRaise_m_BC	77
7.3.4.1.47	tliPrRaise_m_MC	78
7.3.4.1.48	tliPrRaise_c	78
7.3.4.1.49	tliPrRaise_c_BC	79
7.3.4.1.50	tliPrRaise_c_MC	79
7.3.4.1.51	tliPrCatchDetected_m	80
7.3.4.1.52	tliPrCatchDetected_c	80
7.3.4.1.53	tliPrCatchMismatch_m	81
7.3.4.1.54	tliPrCatchMismatch_c	81
7.3.4.1.55	tliPrCatch m	82
7.3.4.1.56	tliPrCatch c	82
7.3.4.1.57	tliPrCatchTimeoutDetected	83
7.3.4.1.58	tliPrCatchTimeout	83
7.3.4.1.59	tliCCreate	83
7.3.4.1.60	tliCStart	84
7.3.4.1.61	tliCRunning	84
7.3.4.1.62	tliCAlive	84
7.3.4.1.63	tliCStop	85
7.3.4.1.64	tliCKill	85
7.3.4.1.65	tliCDoneMismatch	85
7.3.4.1.66	tliCDone	86
7.3.4.1.67	tliCKilledMismatch	86
7.3.4.1.68	tliCKilled	86
7.3.4.1.69	tliCTerminated	87
7.3.4.1.70	tliPConnect	87
7.3.4.1.71	tliPDisconnect	87
7.3.4.1.72	tliPMap	88
7.3.4.1.73	tliPUnmap	88
7.3.4.1.74	tliPClear	88
7.3.4.1.75	tliPStart	89
7.3.4.1.76	tliPStop	89
7.3.4.1.77	tliPHalt	89
7.3.4.1.78	tliEncode	90
7.3.4.1.79	tliDecode	90
7.3.4.1.80	tliTTimeoutDetected	90
7.3.4.1.81	tliTTimeoutMismatch	91
7.3.4.1.82	tliTTimeout	91
7.3.4.1.83	tliTStart	91
7.3.4.1.84	tliTStop	92
7.3.4.1.85	tliTRead	92
7.3.4.1.86	tliTRunning	92
7.3.4.1.87	tliSEnter	93
7.3.4.1.88	tliSLeave	93
7.3.4.1.89	tliVar	93
7.3.4.1.90	tliModulePar	94
7.3.4.1.91	tliGetVerdict	94
7.3.4.1.92	tliSetVerdict	94
7.3.4.1.93	tliLog	95
7.3.4.1.94	tliAEnter	95
7.3.4.1.95	tliALeave	95

7.3.4.1.96	tliANomatch	95
7.3.4.1.97	tliARepeat	96
7.3.4.1.98	tliADefaults	96
7.3.4.1.99	tliAActivate	96
7.3.4.1.100	tliADeactivate	97
7.3.4.1.101	tliAWait	97
7.3.4.1.102	tliAction	97
7.3.4.1.103	tliMatch	97
7.3.4.1.104	tliMatchMismatch	98
7.3.4.1.105	tliInfo	98
8	Java language mapping	98
8.1	Introduction	98
8.2	Names and scopes	98
8.2.1	Names	98
8.2.2	Scopes	99
8.2.2.1	TciParameterType	100
8.2.2.2	TciParameterPassingModeType	100
8.2.2.3	TciParameterListType	100
8.2.2.4	TciTypeClassType	101
8.2.2.5	TciTestComponentKindType	101
8.2.2.6	TciBehaviourIdType	101
8.2.2.7	TciTestCaseIdType	102
8.2.2.8	TciModuleIdType	102
8.2.2.9	TciModuleParameterIdType	102
8.2.2.10	TciModuleParameterListType	102
8.2.2.11	TciModuleParameterType	102
8.2.2.12	TciParameterTypeListType	103
8.2.2.13	TciParameterTypeType	103
8.2.2.14	TciModuleIdListType	103
8.2.3	Abstract type mapping	104
8.2.3.1	Type	104
8.2.4	Abstract value mapping	104
8.2.4.1	Value	105
8.2.4.2	IntegerValue	105
8.2.4.3	FloatValue	105
8.2.4.4	BooleanValue	106
8.2.4.5	CharstringValue	106
8.2.4.6	BitstringValue	106
8.2.4.7	OctetstringValue	107
8.2.4.8	UniversalCharstringValue	108
8.2.4.9	HexstringValue	108
8.2.4.10	RecordValue	109
8.2.4.11	RecordOfValue	110
8.2.4.12	UnionValue	111
8.2.4.13	EnumeratedValue	111
8.2.4.14	VerdictValue	111
8.2.4.15	AddressValue	112
8.2.5	Abstract logging types mapping	112
8.2.5.1	TciValueTemplate	112
8.2.5.2	TciNonValueTemplate	113
8.2.5.3	TciValueList	113
8.2.5.4	TciValueDifference	113
8.2.5.5	TciValueDifferenceList	114
8.3	Constants	114
8.4	Mapping of interfaces	115
8.4.1	The TCI-TM interface	115
8.4.1.1	TCI-TM provided	115
8.4.1.2	TCI-TM required	116
8.4.2	The TCI-CD interface	116
8.4.2.1	TCI-CD provided	116
8.4.2.2	TCI-CD required	116

8.4.3	The TCI-CH interface	116
8.4.3.1	TCI-CH provided	116
8.4.3.2	TCI-CH required	118
8.4.4	The TCI-TL interface.....	119
8.4.4.1	TCI-TL provided.....	119
8.5	Optional parameters	122
8.6	TCI initialization	122
8.7	Error handling	123
9	ANSI C language mapping.....	123
9.1	Introduction	123
9.2	Value interfaces.....	123
9.3	Logging interface	126
9.4	Operation interfaces	127
9.4.1	The TCI-TM interface	127
9.4.1.1	TCI-TM provided.....	127
9.4.1.2	TCI-TM required.....	127
9.4.2	The TCI-CD interface	127
9.4.2.1	TCI-CD provided	127
9.4.2.2	TCI-CD required	128
9.4.3	The TCI-CH interface	128
9.4.3.1	TCI-CH provided	128
9.4.3.2	TCI-CH required	129
9.4.4	The TCI-TL interface.....	129
9.4.4.1	TCI-TL provided.....	129
9.5	Data	133
9.6	Miscellaneous.....	135
9.7	Optional parameters	135
10	C++ language mapping	135
10.1	Introduction	135
10.2	Names and scopes	136
10.3	Memory management.....	136
10.4	Error handling	136
10.5	Type mapping.....	136
10.5.1	Encapsulated C++ types.....	136
10.5.2	General abstract data types	136
10.5.2.1	TciBehaviourId	136
10.5.2.1.1	Methods	137
10.5.2.2	TciModuleId.....	137
10.5.2.2.1	Methods	137
10.5.2.3	TciModuleParameterId	137
10.5.2.3.1	Methods	137
10.5.2.4	TciTestCaselId	138
10.5.2.4.1	Methods	138
10.5.2.5	TciModuleIdList	138
10.5.2.5.1	Methods	138
10.5.2.6	TciModuleParameter	139
10.5.2.6.1	Methods	139
10.5.2.7	TciModuleParameterList.....	139
10.5.2.7.1	Methods	140
10.5.2.8	TciParameterPassingMode.....	140
10.5.2.8.1	Methods	140
10.5.2.8.2	Constants	140
10.5.2.9	TciParameter	141
10.5.2.9.1	Methods	141
10.5.2.10	TciParameterList	141
10.5.2.10.1	Methods	142
10.5.2.11	TciParameterType	142
10.5.2.11.1	Methods	142
10.5.2.12	TciParameterTypeList.....	143
10.5.2.12.1	Methods	143

10.5.2.13	TciTestComponentKind	143
10.5.2.13.1	Methods	144
10.5.2.13.2	Constants	144
10.5.2.14	TciTypeClass	144
10.5.2.14.1	Methods	144
10.5.2.14.2	Constants	145
10.5.3	Abstract TTCN-3 data types and values	145
10.5.3.1	TciType	145
10.5.3.1.1	Methods	146
10.5.3.2	TciValue.....	146
10.5.3.2.1	Methods	146
10.5.3.3	IntegerValue.....	147
10.5.3.3.1	Methods	147
10.5.3.4	FloatValue	147
10.5.3.4.1	Methods	147
10.5.3.5	BooleanValue.....	148
10.5.3.5.1	Methods	148
10.5.3.6	CharstringValue	148
10.5.3.6.1	Methods	148
10.5.3.7	UniversalCharstringValue	149
10.5.3.7.1	Methods	149
10.5.3.8	BitstringValue	150
10.5.3.8.1	Methods	150
10.5.3.9	OctetstringValue	150
10.5.3.9.1	Methods	151
10.5.3.10	HexstringValue	151
10.5.3.10.1	Methods	151
10.5.3.11	RecordValue.....	152
10.5.3.11.1	Methods	152
10.5.3.12	RecordOfValue	152
10.5.3.12.1	Methods	153
10.5.3.13	UnionValue	153
10.5.3.13.1	Methods	153
10.5.3.14	EnumeratedValue.....	154
10.5.3.14.1	Methods	154
10.5.3.15	VerdictValue	154
10.5.3.15.1	Methods	154
10.5.3.16	VerdictValueEnum.....	155
10.5.3.16.1	Methods	155
10.5.3.16.2	Constants	155
10.5.3.17	AddressValue	155
10.5.3.17.1	Methods	155
10.5.4	Abstract logging types	156
10.5.4.1	TciValueTemplate.....	156
10.5.4.1.1	Methods	156
10.5.4.2	TciNonValueTemplate.....	156
10.5.4.2.1	Methods	157
10.5.4.3	TciValueList.....	157
10.5.4.3.1	Methods	157
10.5.4.4	TciValueDifference.....	158
10.5.4.4.1	Methods	158
10.5.4.5	TciValueDifferenceList.....	158
10.5.4.5.1	Methods	159
10.5.4.6	ComponentStatus	159
10.5.4.6.1	Methods	159
10.5.4.6.2	Constants	159
10.5.4.7	TimerStatus	160
10.5.4.7.1	Methods	160
10.5.4.7.2	Constants	160
10.5.4.8	TciStatus	160
10.5.4.8.1	Methods	161
10.5.4.8.2	Constants	161

10.6	Operations mapping	161
10.6.1	TCI-TM	161
10.6.1.1	TciTmRequired	161
10.6.1.2	TciTmProvided	162
10.6.2	TCI-CD	162
10.6.2.1	TciCdRequired	162
10.6.2.2	TciCdProvided	163
10.6.3	TCI-CH	163
10.6.3.1	TciChRequired	163
10.6.3.2	TciChProvided	164
10.6.4	TCI-TL	166
10.6.4.1	TciTlProvided	166
11	W3C XML mapping	174
11.1	Introduction	174
11.2	Scopes	174
11.3	Type mapping	175
11.3.1	Mapping of simple types	175
11.3.1.1	TBoolean	175
11.3.1.2	TString	175
11.3.1.3	TInteger	175
11.3.1.4	TriTimerDurationType	175
11.3.1.5	TciParameterPassingModeType	175
11.3.1.6	TriStatusType	175
11.3.1.7	TciStatusType	175
11.3.1.8	ComponentStatusType	175
11.3.1.9	TimerStatusType	175
11.3.1.10	PortStatusType	175
11.3.2	Complex type mapping	176
11.3.2.1	TriPortIdType	176
11.3.2.2	TriComponentIdType	176
11.3.2.3	TriComponentIdListType	176
11.3.2.4	Port	177
11.3.2.5	Id	177
11.3.2.6	TriMessageType	177
11.3.2.7	TriParameterType	178
11.3.2.8	TriParameterListType	178
11.3.2.9	TriAddressType	178
11.3.2.10	TriAddressListType	178
11.3.2.11	TriExceptionType	179
11.3.2.12	TriSignatureIdType	179
11.3.2.13	TriTimerIdType	179
11.3.2.14	TriTimerDurationType	179
11.3.2.15	QualifiedName	180
11.3.2.16	TciBehaviourIdType	180
11.3.2.17	TciTestCaseIdType	180
11.3.2.18	TciParameterType	180
11.3.2.19	TciParameterListType	181
11.3.3	Abstract value mapping	181
11.3.3.1	Value	181
11.3.3.2	IntegerValue	182
11.3.3.3	FloatValue	183
11.3.3.4	BooleanValue	183
11.3.3.5	Void	183
11.3.3.6	VerdictValue	183
11.3.3.7	BitstringValue	184
11.3.3.8	HexstringValue	184
11.3.3.9	OctetstringValue	184
11.3.3.10	CharstringValue	185
11.3.3.11	UniversalCharstringValue	185
11.3.3.12	RecordValue	186
11.3.3.13	RecordOfValue	187

11.3.3.14	ArrayValue.....	188
11.3.3.15	SetValue.....	189
11.3.3.16	SetOfValue.....	191
11.3.3.17	EnumeratedValue.....	192
11.3.3.18	UnionValue.....	192
11.3.3.19	AnytypeValue.....	193
11.3.3.20	AddressValue.....	194
11.3.4	Abstract logging types mapping.....	195
11.3.4.1	TciValueTemplate.....	195
11.3.4.2	TciNonValueTemplate.....	197
11.3.4.3	TciValueList.....	197
11.3.4.4	TciValueDifference.....	197
11.3.4.5	TciValueDifferenceList.....	198
11.4	Mapping of the operations on the logging interface.....	198
11.4.1	Event.....	198
11.4.2	The TCI-TL interface.....	199
11.4.2.1	TCI-TL provided.....	199
12	Use scenarios.....	217
12.1	Initialization, collecting information, logging.....	217
12.1.1	Use scenario: initialization.....	217
12.1.1.1	Sequence diagram.....	218
12.1.1.2	TTCN-3 fragment.....	218
12.1.2	Use scenario: requesting module parameters.....	218
12.1.2.1	Sequence diagram.....	219
12.1.2.2	TTCN-3 fragment.....	219
12.1.3	Use scenario: logging.....	219
12.1.3.1	Sequence diagram.....	220
12.1.3.2	TTCN-3 fragment.....	220
12.2	Execution of test cases and control.....	220
12.2.1	Use scenario: execution of control.....	220
12.2.1.1	Sequence diagram.....	220
12.2.1.2	TTCN-3 fragment.....	221
12.2.2	Use scenario: test case execution within control.....	221
12.2.2.1	Sequence diagram.....	221
12.2.2.2	TTCN-3 fragment.....	221
12.2.3	Use scenario: direct test case execution.....	221
12.2.3.1	Sequence diagram.....	222
12.2.3.2	TTCN-3 fragment.....	222
12.2.4	Use scenario: execute test case to TRI.....	222
12.2.4.1	Sequence diagram.....	222
12.2.4.2	TTCN-3 fragment.....	223
12.3	Component handling.....	223
12.3.1	Use scenario: local control component creation.....	223
12.3.1.1	Sequence diagram.....	223
12.3.1.2	TTCN-3 fragment.....	224
12.3.2	Use scenario: remote control component creation.....	224
12.3.2.1	Sequence diagram.....	224
12.3.2.2	TTCN-3 fragment.....	224
12.3.3	Use scenario: local MTC creation.....	225
12.3.3.1	Sequence diagram.....	225
12.3.3.2	TTCN-3 fragment.....	225
12.3.4	Use scenario: remote MTC creation.....	225
12.3.4.1	Sequence diagram.....	226
12.3.4.2	TTCN-3 fragment.....	226
12.3.5	Use scenario: component handling for test case execution within control.....	226
12.3.5.1	Sequence diagram.....	227
12.3.5.2	TTCN-3 fragment.....	227
12.3.6	Use scenario: component handling for direct test case execution.....	228
12.3.6.1	Sequence diagram.....	228
12.3.6.2	TTCN-3 fragment.....	229
12.3.7	Use scenario: propagation of map/connect.....	229

12.3.7.1	Sequence diagram	229
12.3.7.2	TTCN-3 fragment	229
12.3.8	Use scenario: propagation of unmap/disconnect.....	230
12.3.8.1	Sequence diagram	230
12.3.8.2	TTCN-3 fragment	230
12.4	Termination of test cases and control	230
12.4.1	Use scenario: stop a test case	230
12.4.1.1	Sequence diagram	231
12.4.1.2	TTCN-3 fragment	231
12.4.2	Use scenario: stop control.....	231
12.4.2.1	Sequence diagram	232
12.4.2.2	TTCN-3 fragment	232
12.4.3	Use scenario: termination of control after error.....	232
12.4.3.1	Sequence diagram	233
12.4.3.2	TTCN-3 fragment	233
12.4.4	Use scenario: termination of a test case after error.....	233
12.4.4.1	Sequence diagram	234
12.4.4.2	TTCN-3 fragment	235
12.4.5	Use scenario: reset	235
12.4.5.1	Sequence diagram	235
12.4.5.2	TTCN-3 fragment	235
12.5	Communication	235
11.5.1	Use scenario: local intercomponent communication	235
12.5.1.1	Sequence diagram	236
12.5.1.2	TTCN-3 fragment	236
12.5.2	Use scenario: internode communication between test components	236
12.5.2.1	Sequence diagram	237
12.5.2.2	TTCN-3 fragment	237
12.5.3	Use scenario: encoding	237
12.5.3.1	Sequence diagram	238
12.5.3.2	TTCN-3 fragment	238
12.5.4	Use scenario: decoding	238
12.5.4.1	Sequence diagram	239
12.5.4.2	TTCN-3 fragment	239
Annex A (normative):	IDL Specification of TCI.....	240
Annex B (normative):	XML Mapping for TCI TL Provided.....	256
B.1	TCI-TL XML Schema for Simple Types	256
B.2	TCI-TL XML Schema for Types	257
B.3	TCI-TL XML Schema for Values	259
B.4	TCI-TL XML Schema for Templates	264
B.5	TCI-TL XML Schema for Events	271
B.6	TCI-TL XML Schema for a Log.....	290
Annex C (informative):	Bibliography.....	293
History		294

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 6 of a multi-part deliverable. Full details of the entire series can be found in part 1 [1].

1 Scope

The present document 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. The present document 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 clause 2). The interface definition uses the CORBA Interface Definition Language (IDL) to specify the TCI completely. Clauses 8, 9, and 9.7 present language mappings for this abstract specification to the target languages Java, ANSI C, and C++. A summary of the IDL-based interface specification is provided in annex A.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
 - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
 - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at <http://docbox.etsi.org/Reference>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

- [1] ETSI ES 201 873-1: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".
- [2] ETSI ES 201 873-4: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics".
- [3] ETSI ES 201 873-5: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".
- [4] ISO/IEC 9646-1: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts".
- [5] ISO/IEC 10646-1: "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".
- [6] W3C Recommendation: "XML Schema Part 0: Primer".

NOTE: See at <http://www.w3.org/TR/xmlschema-0/>.

[7] W3C Recommendation: "XML Schema Part 1: Structures".

NOTE: See at <http://www.w3.org/TR/xmlschema-1/>.

[8] W3C Recommendation: "XML Schema Part 2: Datatypes".

NOTE: See at <http://www.w3.org/TR/xmlschema-2/>.

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Not applicable.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ISO/IEC 9646-1 [4] and the following apply:

Abstract Test Suite (ATS): test suite composed of abstract test cases

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

Coding/Decoding (CD): entity that administers the value and type handling incl. encoding and decoding in the TTCN-3 test system

Component Handling (CH): entity that administers the handling of test components in the TTCN-3 test system

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.

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

Executable Test Suite (ETS): Refer to ISO/IEC 9646-1 [4].

Implementation eXtra Information for Testing (IXIT): Refer to ISO/IEC 9646-1 [4].

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.

real test system interface: Refer to ISO/IEC 9646-1 [4].

System Under Test (SUT): Refer to ISO/IEC 9646-1 [4].

SUT Adaptor (SA): entity that adapts the TTCN-3 communication operations with the SUT based on an abstract test system interface

NOTE: It implements the real test system interface.

Testing and Test Control Notation (TTCN-3): Refer to ISO/IEC 9646-1 [4].

test case: Refer to ISO/IEC 9646-1 [4].

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

Test Logging (TL): entity which provides logging information about test execution (including also the information provided by the TTCN-3 log statement)

Test Management (TM): entity which provides a user interface to as well as the administration of the TTCN-3 test system

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.

test system: Refer to ISO/IEC 9646-1 [4].

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

TTCN-3 Executable (TE): part of a test system that deals with interpretation or execution of a TTCN-3 ETS

TTCN-3 Control Interfaces (TCI): four interfaces that define the interaction of the TTCN-3 Executable with the test management, the coding and decoding, the test component handling, and the logging in a test system

TTCN-3 Runtime Interface (TRI): two interfaces that defines the interaction of the TTCN-3 Executable between the SUT and the Platform Adapter (PA) and the System Adapter (SA) in a test system

3.2 Abbreviations

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

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ATS	Abstract Test Suite
CD	(External) Coding/Decoding
CH	Component Handler
CORBA	Common Object Request Broker Architecture
ETS	Executable Test Suite
IDL	Interface Definition Language
IXIT	Implementation eXtra Information for Testing
MTC	Main Test Component
OMG	Object Management Group
PA	Platform Adaptor
PTC	Parallel Test Component
SA	SUT Adaptor
STL	Standard Template Library of C++
SUT	System Under Test
TCI	TTCN-3 Control Interfaces
TE	TTCN-3 Executable
TL	Test Logging
TM	Test Management
TMC	Test Management and Control
TRI	TTCN-3 Runtime Interfaces
TSI	Test System Interface
TTCN-3	Testing and Test Control Notation Version 3
UML	Unified Modelling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

4 Introduction

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

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

- Test Management and Control (TMC).
- TTCN-3 Executable (TE).
- SUT Adaptor (SA).
- Platform Adaptor (PA).

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

The second part of the present document 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 (ES 201 873-5 [3]).

5 Compliance

The minimum required for a TCI compliant TTCN-3 test system is to adhere to the interface specification stated in the present document. The TTCN-3 semantics in the test system must adhere to the operational semantics defined in ES 201 873-4 [2]. 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 the present document. For the logging interface, the XML mapping can be used instead of the Java or the C mapping.

6 General structure of a TTCN-3 test system

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

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

6.1 Entities in a TTCN-3 test system

The structure of a TTCN-3 test system implementation is illustrated in figure 1.

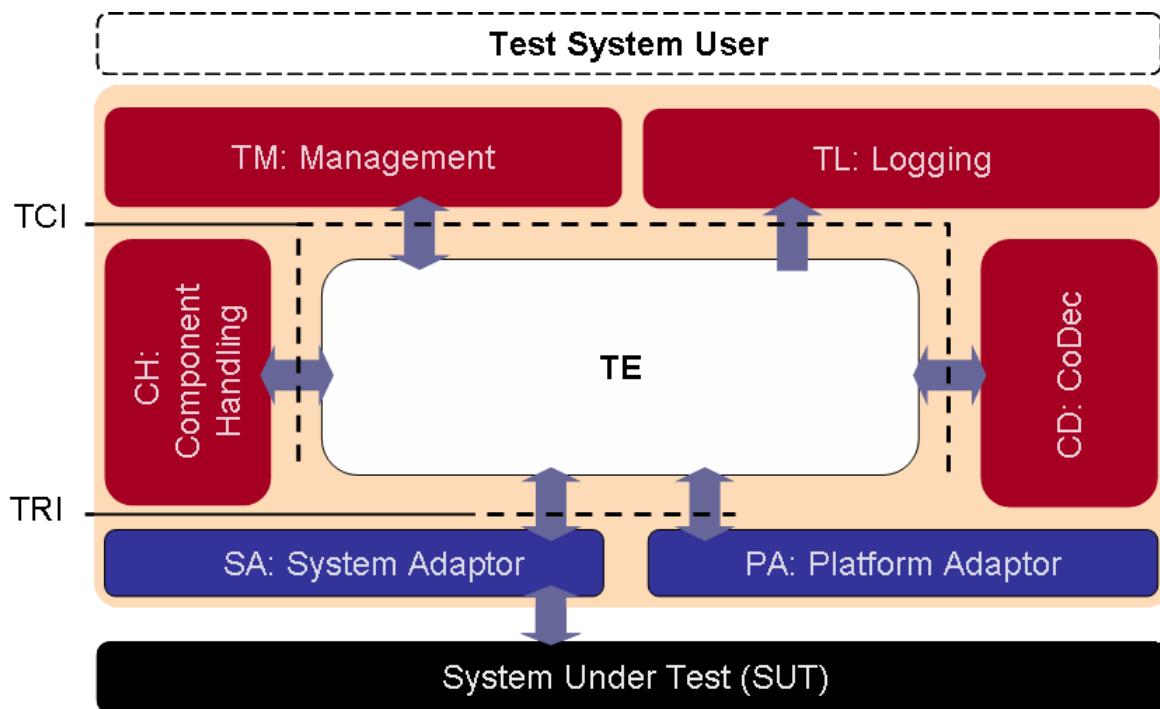


Figure 1: General structure of a TTCN-3 test system

As shown in figure 1, the TTCN-3 Executable (TE), also referred to as the Executable Test Suite (ETS), interprets and executes TTCN-3 modules. Various TE structural elements can be identified: control, behaviour, components, types, values and queues. The structural elements within the TE represent functionality that is defined within a TTCN-3 module or by the TTCN-3 standard (ES 201 873-1 [1]) 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 ES 201 873-5 [3]. The TCI specification defines the interaction between the TE and the TMC.

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

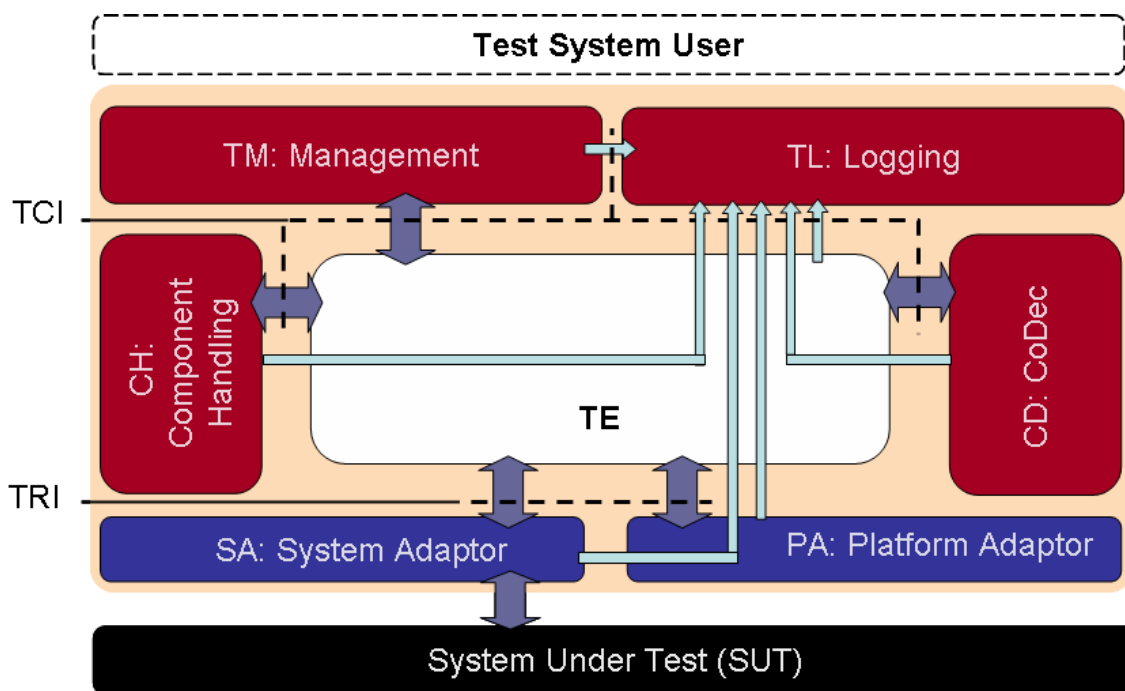


Figure 2: Detailed View on TL

6.1.1 Test Management and Control (TMC)

The TMC entity includes functionality related to management of:

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

6.1.1.1 Test Management (TM)

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

6.1.1.2 Coding and Decoding (CD)

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

6.1.1.3 Component Handling (CH)

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

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

The general structure of a test system distributed among several nodes is depicted in figure 3.

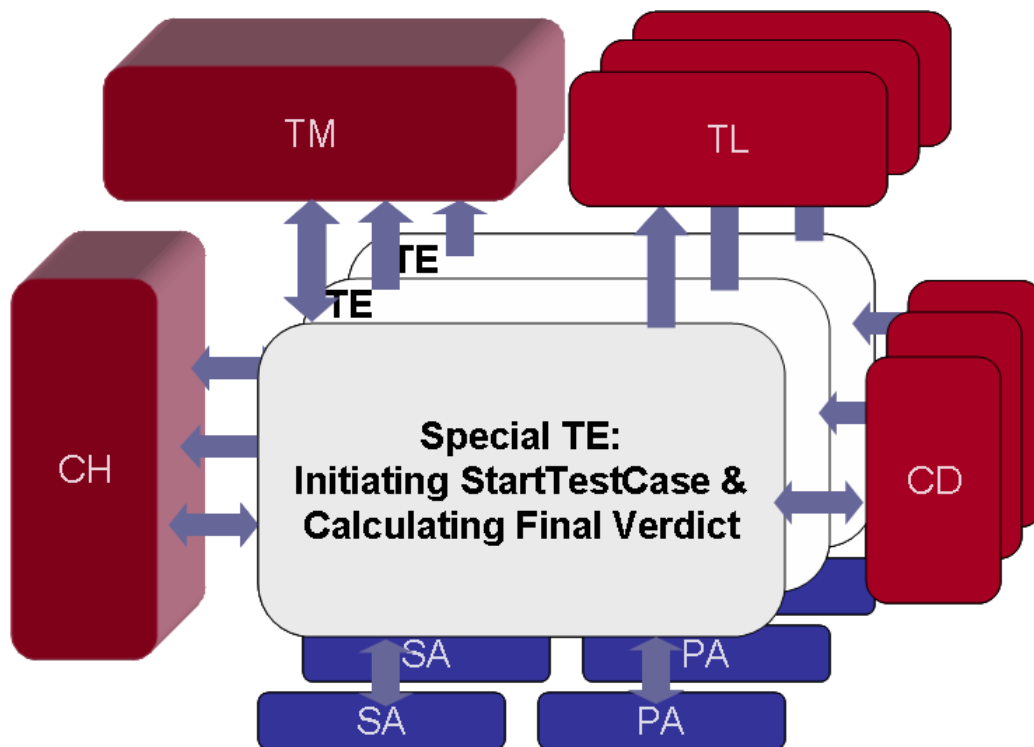


Figure 3: General structure of a distributed TTCN-3 test system

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

NOTE 2: As stated in ES 201 873-4 [2], a test system executes at most one test case at a given point in time, i.e. a TTCN-3 module cannot execute multiple test cases at the same time.

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

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

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

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

Additional communication is needed to implement the distribution of test components onto several nodes. Component management communication includes the indication of the creation of test components, the starting of execution of a test component, verdict distribution, as well as component termination indication. The CH does not implement the TTCN-3 component behaviour. Rather, it implements the communication between several components implemented by a TE.

6.1.1.4 Test Logging (TL)

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

6.1.2 TTCN-3 Executable (TE)

The TE entity executes or interprets a TTCN-3 module. Conceptually, the TE can be decomposed into six interacting entities: a Control, Behaviour, Component, Type, Value, and Queue entity. This structural decomposition of the TE is defined in ES 201 873-5 [3]. The terminology for TE defined in ES 201 873-5 [3] is used within the present document.

6.1.3 SUT Adaptor (SA)

The SA is the implementation of the System under Test Adaptor (SA) as defined in ES 201 873-5 [3]. The terminology for SA defined in ES 201 873-5 [3] is used within the present document.

6.1.4 Platform Adaptor (PA)

The PA is the implementation of the Platform Adaptor (PA) as defined in ES 201 873-5 [3]. The terminology for PA defined in ES 201 873-5 [3] is used within the present document.

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.
- **TCI 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 bi-directional 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 ES 201 873-5 [3].

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

TTCN-3 Operation Name	TCI Operation Name	TCI Interface Name
send	tciSendConnected (see note 1)	TCI-CH Provided
	tciSendConnectedBC (see note 2)	
	tciSendConnectedMC (see note 3)	
	tciEnqueueMsgConnected	TCI-CH Required
call	tciCallConnected (see note 1)	TCI-CH Provided
	tciCallConnectedBC (see note 2)	
	tciCallConnectedMC (see note 3)	
	tciEnqueueCallConnected	TCI-CH Required
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

TTCN-3 Operation Name	TCI Operation Name	TCI Interface Name
connect	tciConnectReq	TCI-CH Provided
	tciConnect	TCI-CH Required
disconnect	tciDisconnectReq	TCI-CH Provided
	tciDisconnect	TCI-CH Required
map	tciMapReq	TCI-CH Provided
	tciMap	TCI-CH Required
unmap	tciUnmapReq	TCI-CH Provided
	tciUnmap	TCI-CH Required
running	tciTestComponentRunningReq	TCI-CH Provided
	tciTestComponentRunning	TCI-CH Required
alive	tciTestComponentAliveReq	TCI-CH Provided
	tciTestComponentAlive	TCI-CH Required
done	tciTestComponentDoneReq	TCI-CH Provided
	tciTestComponentDone	TCI-CH Required
killed	tciTestComponentKilledReq	TCI-CH Provided
	tciTestComponentKilled	TCI-CH Required
mtc	tciGetMTCReq	TCI-CH Provided
	tciGetMTC	TCI-CH Required
execute	tciTestCaseExecuteReq	TCI-CH Provided
	tciTestCaseExecute	TCI-CH Required
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, 9, and 9.7.

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 <code>Value</code> , which can be absent, 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. CONTROL, MTC, PTC, SYSTEM, and PTC_ALIVE. 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

<code>TciBehaviourIdType</code>	A value of type <code>TciBehaviourIdType</code> identifies a TTCN-3 behaviour functions.
---------------------------------	--

Additional abstract data types with the prefix `Tri` are taken from ES 201 873-5 [3]: `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, 9, and 9.7. 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 the present document 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. <code>boolean</code> , <code>integer</code> , 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>ARRAY</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>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.

NOTE: Newly created instances of empty record types are considered to be initialized.

<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 the present document, TTCN-3 values are represented at the TCI interfaces via numerous abstract data types.

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

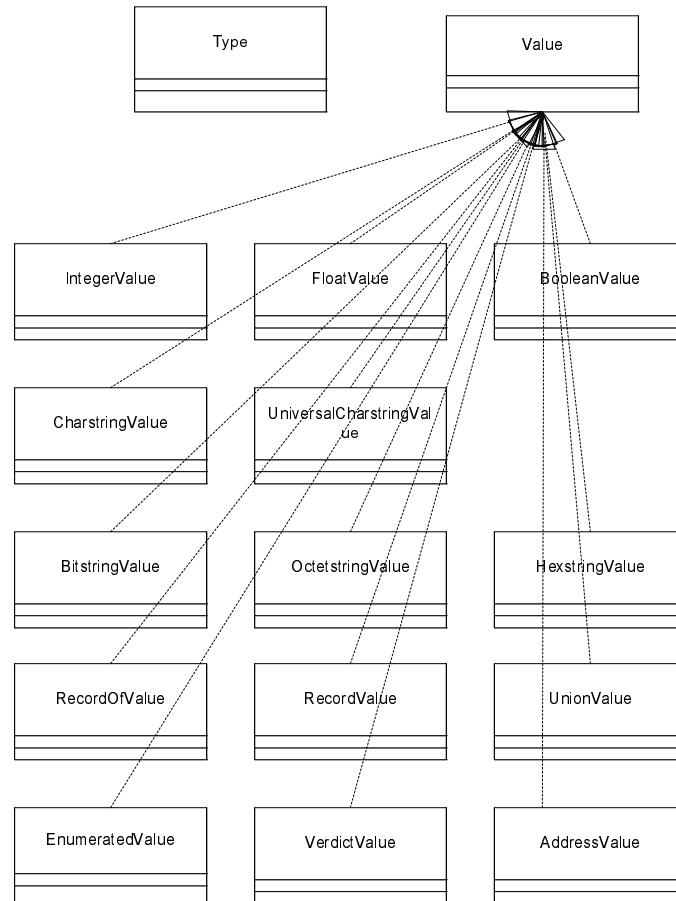


Figure 4: Hierarchy of abstract values

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

7.2.2.2.1 The abstract data type `Value`

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

<code>Type getType()</code>	Returns the type of the specified value.
<code>TBoolean 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`:

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

7.2.2.2.3 The abstract data type FloatValue

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

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

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

7.2.2.2.4 The abstract data type BooleanValue

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

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

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

7.2.2.2.5 The abstract data type CharstringValue

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

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

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

7.2.2.2.6 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 <code>value</code> .
<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 <code>position</code> are from 0 to <code>length - 1</code> .
<code>void setChar(in TInteger position, in TUniversalChar value)</code>	Sets the universal char at position to <code>value</code> . Valid values for <code>position</code> are from 0 to <code>length - 1</code> .
<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 <code>len</code> .

7.2.2.2.7 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 <code>value</code> . E.g. the value of this <code>BitstringValue</code> is 0101 if the textual representation in <code>value</code> 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 <code>position</code> are from 0 to <code>length - 1</code> .
<code>void setBit(in TInteger position, in 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 <code>position</code> are from 0 to <code>length - 1</code> .
<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 <code>len</code> .

7.2.2.2.8 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"</code> . The textual representation of the empty TTCN-3 <code>octetstring</code> is <code>""</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"</code> .
<code>TChar getOctet(in TInteger position)</code>	Returns the value (0..255) at <code>position</code> of this TTCN-3 <code>octetstring</code> . Position 0 denotes the first octet of the TTCN-3 <code>octetstring</code> . Valid values for <code>position</code> are from 0 to <code>length - 1</code> .
<code>void setOctet(in TInteger position, in TInteger value)</code>	Sets the octet at <code>position</code> to value (0..255). Position 0 denotes the first octet in the <code>octetstring</code> . Valid values for <code>position</code> 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> .

7.2.2.2.9 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"</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"</code> .
<code>TChar getHex(in TInteger position)</code>	Returns the value (0..15) at <code>position</code> of this TTCN-3 <code>hexstring</code> . Position 0 denotes the first hex digits of the TTCN-3 <code>hexstring</code> . Valid values for <code>position</code> are from 0 to <code>length - 1</code> .
<code>void setHex(in TInteger position, in TInteger value)</code>	Sets the hex digit at <code>position</code> to value (0..15). Position 0 denotes the first octet in the <code>hexstring</code> . Valid values for <code>position</code> 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> .

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

7.2.2.2.10 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.

NOTE: Newly created instances of empty record types are considered to be initialized.

The same abstract data type applies for values whose type class is `SET`. The distinction between `record` and `set` is only relevant at matching time.

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

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

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

`TStringSeq getFieldNames()` Returns a sequence of string of field names, the empty sequence, if the record has no fields.

`void setFieldOmitted(in TString fieldName)`
Mark the referenced field of the record as being omitted.

7.2.2.2.11 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 `ARRAY` or `SET_OF`. The distinction between `record of`, `set of`, and `array` 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. Also for array values indices start from 0, independent of the lower index bound. The return value is the common abstract base type `Value`, as a `record of` can have fields of any type defined in TTCN-3.

`void setField(in TInteger position, in Value value)`
Sets the field at `position` to `value`. If `position` is greater than `(length - 1)` the record of is extended to have the length `(position + 1)`. The record of elements between the original position at `length` and `position - 1` is set to omit. No assumption shall be made on how a field is stored in a `record of`. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value is copied. Therefore, it should be assumed that subsequent modifications of `value` will not be considered in the `record of`. Also for array values indices start from 0, independent of the lower index bound.

<code>void appendField(in Value value)</code>	Appends the value at the end of the record of, i.e. at position length. No assumption shall be made on how a field is stored in a record of. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value is copied. Therefore, it should be assumed that subsequent modifications of value will not be considered in the record of.
<code>Type getElementType()</code>	Returns the Type of the elements of this record of.
<code>TInteger getLength()</code>	Returns the actual length of the record of value, zero if the record of value is omit.
<code>void setLength(in TInteger len)</code>	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 or equal than the original length this operation is ignored.
<code>TInteger getOffset()</code>	Returns the lowest possible index. For a record of or set of value this is always 0. For an array value, this is the lower index bound used in the type definition.

7.2.2.2.12 The abstract data type UnionValue

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

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

<code>Value getVariant(in TString variantName)</code>	Returns the value of the TTCN-3 union variantName, if variantName equals the result of <code>getPresentVariantName</code> , the distinct value null otherwise. variantName denotes the name of the union variant as defined in TTCN-3.
<code>void setVariant(in TString variantName, in Value value)</code>	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.
<code>TString getPresentVariantName()</code>	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.
<code>TStringSeq getVariantNames()</code>	Returns a sequence of string of variant names, the distinct value null, if the union has no fields. If the <code>UnionValue</code> represents the TTCN-3 anytype, i.e. the type class of the type obtained by <code>getType()</code> is <code>ANYTYPE</code> , all predefined and user-defined TTCN-3 types is returned.

7.2.2.2.13 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`:

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

7.2.2.2.14 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`:

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

7.2.2.2.15 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:

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

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:

<code>TBoolean</code> <code>isOmit()</code>	Returns <code>true</code> if the template is an omit template.
<code>TBoolean</code> <code>isAny()</code>	Returns <code>true</code> if the template is an any template.
<code>TBoolean</code> <code>isAnyOrOmit()</code>	Returns <code>true</code> if the template is an any or omit template.
<code>TString</code> <code>getTemplateDef()</code>	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>TBoolean</code> <code>isAny()</code>	Returns <code>true</code> if the template is an any template.
<code>TBoolean</code> <code>isAll()</code>	Returns <code>true</code> if the template is an all template.
<code>TString</code> <code>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.

`TciValueDifferenceList` A value of `TciValueDifferenceList` is a sequence of value differences.

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

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

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

`Value get(in TInteger index)` Returns the value at the specified position.

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

`Value getValue()` Returns the value of the `TciValueDifference`.

`TciValueTemplate getTciValueTemplate()` Returns the template of the `TciValueDifference`.

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

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

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

`TBoolean isEmpty()` Returns `true` if the list contains no values.

`TciValueDifference get(in TInteger index)` Returns the `TciValueDifference` at the specified position.

7.2.3.4 The Status Types

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

`ComponentStatusType` A value of `ComponentStatusType` is either "inactiveC", "runningC", "stoppedC", or "killedC".

`TimerStatusType` A value of `TimerStatusType` is either "runningT", "inactiveT", or "expiredT".

`PortStatusType` A value of `PortStatusType` is either "startedP", "haltedP", or "stoppedP".

7.3 TCI operations

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

The terms "required" and "provided" reflect the fact that the present document 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, 9, and 9.7. Annex B provides for the logging interface an alternative mapping to XML.

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

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

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

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

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

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

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

7.3.1 The TCI-TM interface

The TCI Test Management Interface (TCI-TM) describes the operations a TTCN-3 Executable is required to implement and the operations a test management implementation shall provide to the TE (figure 5).

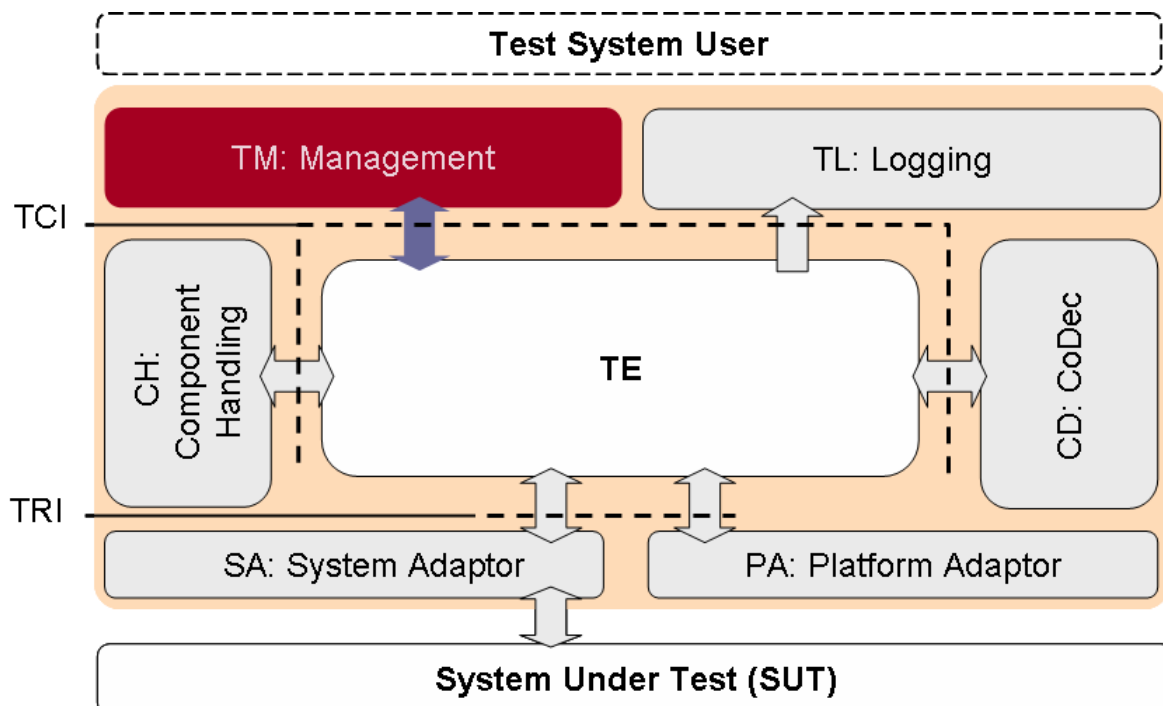


Figure 5: The TCI-TM interface

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

Clause 12 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 tciGetImportedModules

Signature	TciModuleIdListType tciGetImportedModules()	
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 <i>provided</i> 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 <i>provided</i> operation tciControlPartTerminated.

7.3.1.1.10 tciStopControl

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

7.3.1.2 TCI-TM provided

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

7.3.1.2.1 tciTestCaseStarted

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

7.3.1.2.2 tciTestCaseTerminated

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

7.3.1.2.3 tciControlTerminated

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

7.3.1.2.4 tciGetModulePar

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

7.3.1.2.5 tciLog

Signature	void tciLog (in TriComponentIdType testComponentId, in TString message)
In Parameters	testComponentId Identifier of the component that logs the message. message A string value, i.e. the message to be logged.
Return Value	void
Constraint	Shall be called by the TE when the TTCN-3 statement log will be executed, either in the control part of a module or within the test case.
Effect	The TM presents testComponentId and message to the user, how this done is not within the scope of the present document.

7.3.1.2.6 tciError

Signature	void tciError(in TString message)
In Parameters	message A string value, i.e. the error message.
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 (see figure 6).

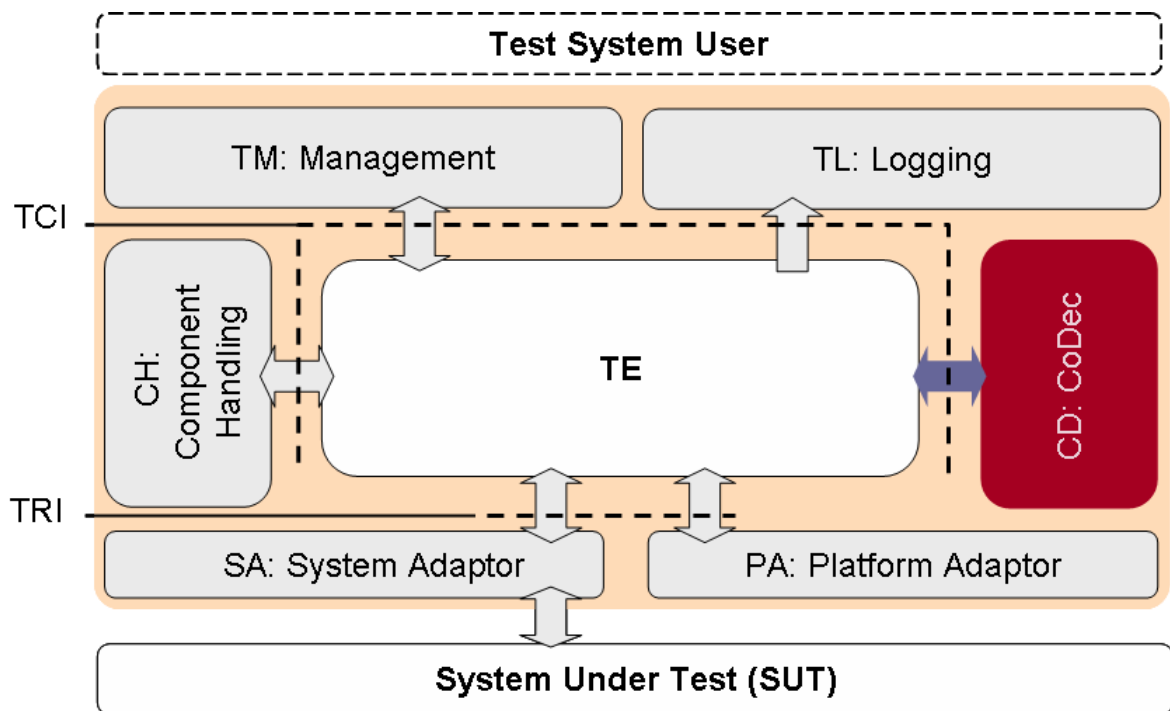


Figure 6: The TCI-CD interface

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

Clause 12 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 <code>getTypeForName</code> (in TString <code>typeName</code>)	
In Parameters	<code>typeName</code>	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" <code>typeName</code> has to be the fully qualified type name, i.e. <code>module.typeName</code>
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 <code>typeName</code> denotes to the basic TTCN-3 type like "charstring", "bitstring" etc. Returns the distinct value <code>null</code> if the requested type cannot be returned. Note that the <code>anytype</code> and <code>address</code> cannot be obtained with <code>module</code> set to <code>null</code> . Although they are predefined types they might be distinct between modules. For example, <code>address</code> 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	<code>Type getInteger()</code>
Return Value	An instance of <code>Type</code> representing a TTCN-3 integer type.
Effect	Constructs and returns a basic TTCN-3 integer type.

7.3.2.1.3 `getFloat`

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

7.3.2.1.4 `getBoolean`

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

7.3.2.1.5 `Void`7.3.2.1.6 `getCharstring`

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

7.3.2.1.7 `getUniversalCharstring`

Signature	<code>Type getUniversalCharstring ()</code>
Return Value	An instance of <code>Type</code> 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	<code>Type getHexstring ()</code>
Return Value	An instance of <code>Type</code> representing a TTCN-3 hexstring type.
Effect	Constructs and returns a basic TTCN-3 hexstring type.

7.3.2.1.9 `getBitstring`

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

7.3.2.1.10 `getOctetstring`

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

7.3.2.1.11 getVerdict

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

7.3.2.1.12 tciErrorReq

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

7.3.2.2 TCI-CD provided

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

7.3.2.2.1 decode

Signature	Value decode(in TriMessageType message, in Type decodingHypothesis)
In Parameters	message The encoded message to be decoded. decodingHypothesis The hypothesis the decoding can be based on.
Return Value	Returns the decoded value, if the value is of a compatible type as the decodingHypothesis, else the distinct value null.
Constraint	This operation shall be called whenever the TE has to decode an encoded value. The TE might decode immediately after reception of an encoded value, or might for performance considerations postpone the decoding until the actual access of the encoded value.
Effect	This operation 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).

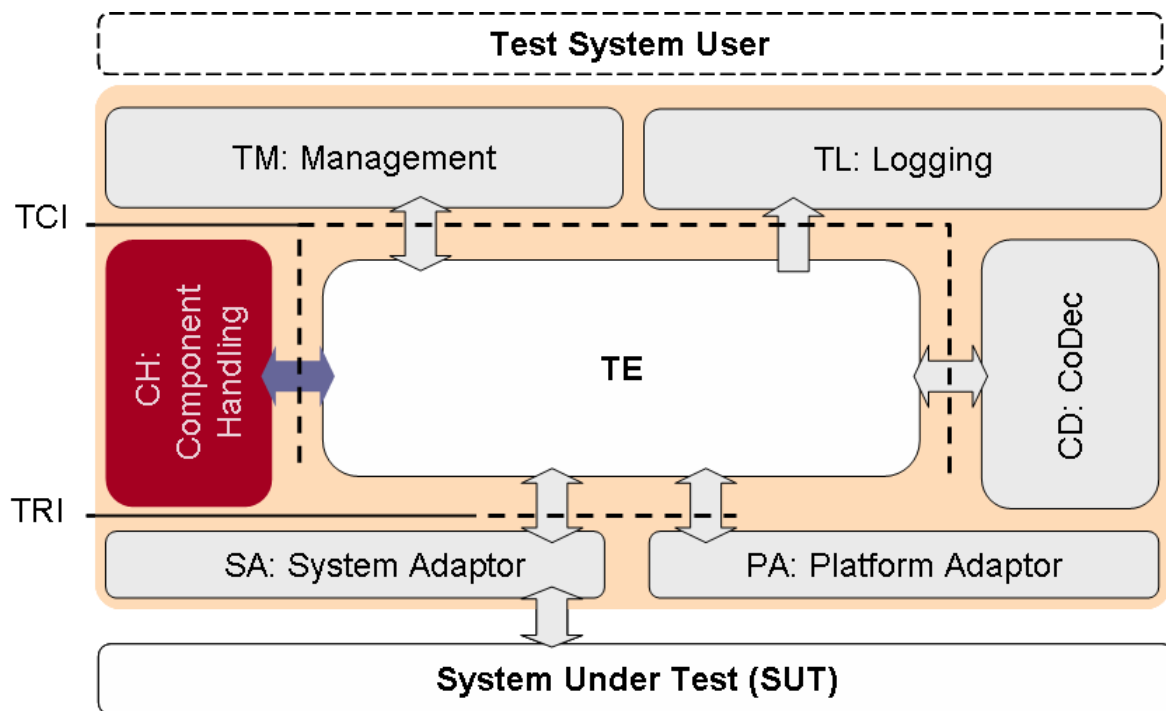


Figure 7: The TCI-CH interface

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

Clause 12 illustrates the usage and sequential ordering of operation calls by either the TE or the CH.

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

7.3.3.1 TCI-CH required

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

7.3.3.1.1 tciEnqueueMsgConnected

Signature	void tciEnqueueMsgConnected (in TriPortIdType sender, in TriComponentIdType receiver, in Value rcvdMessage)	
In Parameters	sender	Port identifier at the sending component via which the message is sent.
	receiver	Identifier of the receiving component.
	rcvdMessage	The value to be enqueued.
Return Value	void	
Constraint	This operation shall be called by the CH at the local TE when at remote TE a <i>provided</i> tciSendConnected has been called.	
Effect	The TE enqueues the received value into the local port queue of the indicated receiver component.	

7.3.3.1.2 tciEnqueueCallConnected

Signature	void tciEnqueueCallConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in TciParameterListType parameterList)	
In Parameters	sender	Port identifier at the sending component via which the message is sent.
	receiver	Identifier of the receiving component.
	signature	Identifier of the signature of the procedure call.
	parameterList	A list of value parameters which are part of the indicated signature. The parameters in parameterList are ordered as they appear in the TTCN-3 signature declaration.
Return Value	void	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a <i>provided</i> tciCallConnected 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 null because they are only of relevance in a reply to the procedure call but not in the procedure call itself. The procedure parameters are the parameters specified in the TTCN-3 signature template.	
Effect	The TE enqueues the calls at the local port queue of the indicated receiver component.	

7.3.3.1.3 tciEnqueueReplyConnected

Signature	void tciEnqueueReplyConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in TciParameterListType parameterList, in Value returnValue)	
In Parameters	sender	Identifier of the port sending the reply.
	receiver	Identifier of the component receiving the reply.
	signature	Identifier of the signature of the procedure call.
	parameterList	A list of value parameters which are part of the indicated signature. The parameters in parameterList are ordered as they appear in the TTCN-3 signature declaration.
	returnValue	(Optional) return value of the procedure call.
Return Value	void	
Constraint	This operation shall be called by the CH at the local TE when at a remote TE a <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 (any kind except of SYSTEM).
	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 TciParameterListType 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.	
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. Because the out values of <i>inout</i> and <i>out</i> parameters of a function being executed on a test component have no effect on that test component (ES 201 873-1 [1]), 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 being returned. If the MTC is not executed on the local TE the distinct value null will be returned. The operation will have no effect on the execution of the MTC. After the operation returns, the CH will communicate the value back to the remote TE.	

7.3.3.1.16 tciExecuteTestCase

Signature	void tciExecuteTestCase (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)	
In Parameters	testCaseId	A test case identifier as defined in the TTCN-3 module.
	tsiPortList	Contains all ports that have been declared in the definition of the system component for the test case, i.e. the TSI ports. If a system component has not been explicitly defined for the test case, then the tsiPortList contains all communication ports of the MTC. The ports in tsiPortList are ordered as they appear in the respective TTCN-3 component type declaration. If no ports have to be passed either the null value or an empty tsiPortList, i.e. a list of length zero shall be passed.
Return Value	void	
Constraint	This operation shall be called by the CH at the appropriate local TE when at a remote TE a <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 send.
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 send.
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 send.
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 <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	On invocation of this operation the TE can initiate the procedure call corresponding to the signature identifier <i>signature</i> at the called component receiver. The <code>tciCallConnected</code> 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 <code>tciCallConnected</code> 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. <code>triStartTimer</code> . 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 <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	On invocation of this operation the TE can initiate the procedure call corresponding to the signature identifier <i>signature</i> at the called component receiver. The <code>tciCallConnected</code> 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 <code>tciCallConnected</code> 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. <code>triStartTimer</code> . 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 <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 component identifier <code>receiver</code> . CH transmits the reply to the remote TE on which <code>receiver</code> 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 <code>signature</code> and component identifier <code>receiver</code> . CH transmits the exception to the remote TE on which <code>receiver</code> 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 <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.	

7.3.3.2.12 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 <code>signature</code> and one of the component identifier <code>receivers</code> . CH transmits the exception to all remote TEs on which <code>receivers</code> 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 (any kind except of SYSTEM).
	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. <code>name</code> shall be set to the distinct value <code>null</code> 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 <code>tciCreateTestComponent</code> 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 TciParameterListType 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 <code>parameterList</code> are ordered as they appear in the TTCN-3 signature of the test case. If no parameters have to be passed either the <code>null</code> value or an empty <code>parameterList</code> , 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.	
Effect	CH transmits the start component request to the remote TE and calls there the <code>tciStartTestComponent</code> 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 <code>tciStopTestComponent</code> 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 TTCN-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. Because the out values of <i>inout</i> and <i>out</i> parameters of a function being executed on a test component have no effect on that test component (ES 201 873-1 [1]), the tciTestComponentTerminateReq operation does not have a parameterList parameter. CH communicates the termination of the indicated component to all participating TEs and to the special TE*, which keeps track of the overall verdict.	

7.3.3.2.21 tciTestComponentRunningReq

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

7.3.3.2.22 tciTestComponentDoneReq

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

7.3.3.2.23 tciGetMTCReq

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

7.3.3.2.24 tciExecuteTestCaseReq

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

7.3.3.2.25 tciResetReq

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

7.3.3.2.26 tciKillTestComponentReq

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

7.3.3.2.27 tciTestComponentAliveReq

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

7.3.3.2.28 tciTestComponentKilledReq

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

7.3.4 The TCI-TL interface

The TCI Test Logging Interface (TCI-TL) describes the operations a TTCN-3 Executable is required to implement and the operations a test logging implementation shall provide to the TE (figure 8).

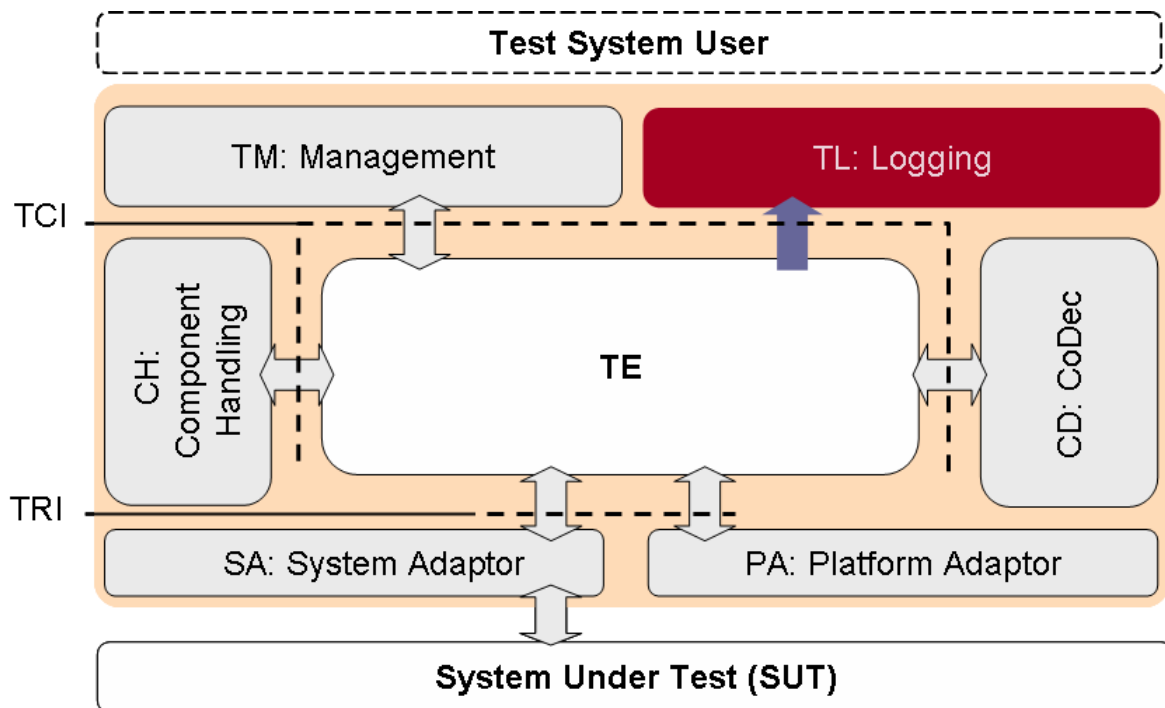


Figure 8: The TCI-TL interface

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

7.3.4.1 TCI-TL provided

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

7.3.4.1.1 tliTcExecute

Signature	void tliTcExecute(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TciParameterListType triPars, in TriTimerDurationType dur)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	tcId	The testcase to be executed.
	triPars	The list of parameters required by the testcase.
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 the present document.	

7.3.4.1.2 tliTcStart

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

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 the present document.	

7.3.4.1.4 tliTcStarted

Signature	void tliTcStarted(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TciParameterListType tciPars, in TriTimerDurationType dur)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	tcId	The testcase to be executed.
	tciPars	The list of parameters required by the testcase.
dur	Duration of the execution.	
Return Value	Void	
Constraint	Shall be called by TM or TE 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 the present document.	

7.3.4.1.5 tliTcTerminated

Signature	void tliTcTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TciParameterListType tciPars, in VerdictValue verdict)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	tcId	The testcase to be executed.
	tciPars	The list of parameters required by the testcase.
verdict	The verdict of the testcase.	
Return Value	void	
Constraint	Shall be called by TM or TE 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 the present document.	

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 the present document.	

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 the present document.	

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 or TE 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 the present document.	

7.3.4.1.9 tliMSend_m

Signature	void tliMSend_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue, in Value addrValue, in TciStatusType encoderFailure, in TriMessageType msg, in TriAddressType address, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	to	The port to which the message is sent.
	msgValue	The value to be encoded and sent.
	addrValue	The address value of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded message.
	address	The address of the destination within the SUT.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA or TE 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 the present document.	

7.3.4.1.10 tliMSend_m_BC

Signature	void tliMSend_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue, in TciStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	to	The port to which the message is sent.
	msgValue	The value to be encoded and sent.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded message.
transmissionFailure	The failure message which might occur at transmission.	
Return Value	void	
Constraint	Shall be called by SA or TE 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 the present document.	

7.3.4.1.11 tliMSend_m_MC

Signature	void tliMSend_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue, in TciValueList addrValues, in TciStatusType encoderFailure, in TriMessageType msg, in TriAddressListType addresses, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is sent.
	to	The port to which the message is sent.
	msgValue	The value to be encoded and sent.
	addrValues	The address values of the destinations within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	msg	The encoded message.
	addresses	The addresses of the destinations within the SUT.
	transmissionFailure	The failure message which might occur at transmission.
Return Value	void	
Constraint	Shall be called by SA or TE 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 the present document.	

7.3.4.1.12 tliMSend_c

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

7.3.4.1.13 tliMSend_c_BC

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

7.3.4.1.14 tliMSend_c_MC

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

7.3.4.1.15 tliMDetected_m

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

7.3.4.1.16 tliMDetected_c

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

7.3.4.1.17 tliMMismatch_m

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

7.3.4.1.18 tliMMismatch_c

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

7.3.4.1.19 tliMReceive_m

Signature	void tliMReceive_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in Value msgValue, in TciValueTemplate msgTpl, in Value addrValue, in TciValueTemplate addressTpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	msgValue	The message which is checked against the template.
	msgTpl	The template used to check the message match.
	addrValue	The address value of the source within the SUT.
	addressTpl	The expected address of the source within the SUT.
Return Value	Void	
Constraint	Shall be called by TE to log the receiving of a message. This event occurs after checking a template match. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.20 tliMReceive_c

Signature	void tliMReceive_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in Value msgValue, in TciValueTemplate msgTpl, in TriComponentIdType from, in TciNonValueTemplate fromTpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the message is received.
	msg	The message which is checked against the template.
	msgTpl	The template used to check the message match.
	from	The component which sent the message.
	fromTpl	The expected sender component.
Return Value	Void	
Constraint	Shall be called by TE to log the receive of a message. This event occurs after checking a template match. This event is used for logging the intercomponent communication.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.21 tliPrCall_m

Signature	void tliPrCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value addrValue, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriAddressType address, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	addrValue	The address value of the destination within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	address	The address of the destination within the SUT.
transmissionFailure	The failure message which might occur at transmission.	
Return Value	Void	
Constraint	Shall be called by SA or TE to log a unicast call operation. This event occurs after call execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.22 tliPrCall_m_BC

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

7.3.4.1.23 tliPrCall_m_MC

Signature	void tliPrCall_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in TciValueList addrValues, in TciStatusType encoderFailure, in TriParameterListType triPars, in TriAddressListType addresses, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is invoked.
	to	The port to which the call is sent.
	signature	The signature of the called operation.
	tciPars	The parameters of the called operation.
	addrValues	The address values of the destinations within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	triPars	The encoded parameters.
	addresses	The addresses of the destinations within the SUT.
transmissionFailure	The failure message which might occur at transmission.	
Return Value	Void	
Constraint	Shall be called by SA or TE to log a multicast call operation. This event occurs after call execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.24 tliPrCall_c

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

7.3.4.1.25 tliPrCall_c_BC

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

7.3.4.1.26 tliPrCall_c_MC

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

7.3.4.1.27 tliPrGetCallDetected_m

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

7.3.4.1.28 tliPrGetCallDetected_c

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

7.3.4.1.29 tliPrGetCallMismatch_m

Signature	void tliPrGetCallMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in TciParameterListType tciPars, in TciValueTemplate parsTpl, in TciValueDifferenceList diffs, in Value addrValue, in TciValueTemplate addressTpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the call is received.
	signature	The signature of the detected call.
	tciPars	The parameters of detected call.
	parsTpl	The template used to check the parameter match.
	diffs	The difference/the mismatch between call and template
	addrValue	The address value of the source within the SUT.
	addressTpl	The expected address of the source within the SUT.
Return Value	Void	
Constraint	Shall be called by TE to log the mismatch of a getcall. This event occurs after getcall is checked against a template. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.30 tliPrGetCallMismatch_c

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

7.3.4.1.31 tliPrGetCall_m

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

7.3.4.1.32 tliPrGetCall_c

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

7.3.4.1.33 tliPrReply_m

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

7.3.4.1.34 tliPrReply_m_BC

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

7.3.4.1.35 tliPrReply_m_MC

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

7.3.4.1.36 tliPrReply_c

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

7.3.4.1.37 tliPrReply_c_BC

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

7.3.4.1.38 tliPrReply_c_MC

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

7.3.4.1.39 tliPrGetReplyDetected_m

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

7.3.4.1.40 tliPrGetReplyDetected_c

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

7.3.4.1.41 tliPrGetReplyMismatch_m

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

7.3.4.1.42 tliPrGetReplyMismatch_c

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

7.3.4.1.43 tliPrGetReply_m

Signature	void tliPrGetReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in TciParameterListType tciPars, in TciValueTemplate parsTmpl, in Value replValue, in TciValueTemplate replyTmpl, in Value addrValue, in TciValueTemplate addressTmpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the reply is received.
	signature	The signature relating to the reply.
	tciPars	The signature parameters relating to the reply.
	parsTmpl	The template used to check the parameter match.
	replValue	The received reply.
	replyTmpl	The template used to check the reply match.
	addrValue	The address value of the source within the SUT.
	addressTmpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by TE to log getting a reply. This event occurs after getreply is checked against a template. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.44 tliPrGetReply_c

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

7.3.4.1.45 tliPrRaise_m

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

7.3.4.1.46 tliPrRaise_m_BC

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

7.3.4.1.47 tliPrRaise_m_MC

Signature	void tliPrRaise_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in TriSignatureIdType signature, in TciParameterListType tciPars, in Value excValue, in TciValueList addrValues, in TciStatusType encoderFailure, in TriExceptionType exc, in TriAddressListType addresses, in TriStatusType transmissionFailure)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is sent.
	to	The port to which the exception is sent.
	signature	The signature relating to the exception.
	tciPars	The signature parameters relating to the exception.
	excValue	The exception to be sent.
	addrValues	The address values of the destinations within the SUT.
	encoderFailure	The failure message which might occur at encoding.
	exc	The encoded exception.
	addresses	The addresses of the destinations within the SUT.
transmissionFailure	The failure message which might occur at transmission.	
Return Value	void	
Constraint	Shall be called by SA or TE to log a multicast raise operation. This event occurs after reply execution. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.48 tliPrRaise_c

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

7.3.4.1.49 tliPrRaise_c_BC

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

7.3.4.1.50 tliPrRaise_c_MC

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

7.3.4.1.51 tliPrCatchDetected_m

Signature	void tliPrCatchDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	from	The port from which the exception has been sent.
	signature	The signature relating to the exception.
	exc	The exception caught.
Return Value	void	
Constraint	Shall be called by SA or TE to log the catch enqueue operation. This event occurs after catch is enqueued. This event is used for logging the communication with the SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.52 tliPrCatchDetected_c

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

7.3.4.1.53 tliPrCatchMismatch_m

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

7.3.4.1.54 tliPrCatchMismatch_c

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

7.3.4.1.55 tliPrCatch m

Signature	void tliPrCatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature, in Value excValue, in TciValueTemplate excTpl, in Value addrValue, in TciValueTemplate addressTpl)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
	excValue	The received exception.
	excTpl	The template used to check the exception match.
	addrValue	The address value of the source within the SUT.
	addressTpl	The expected address of the source within the SUT.
Return Value	void	
Constraint	Shall be called by SA or TE to log catching an exception. This event occurs after catch is checked against a template. This event is used for logging the communication with SUT.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.56 tliPrCatch c

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

7.3.4.1.57 tliPrCatchTimeoutDetected

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

7.3.4.1.58 tliPrCatchTimeout

Signature	void tliPrCatchTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType at, in TriSignatureIdType signature)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	at	The port via which the exception is received.
	signature	The signature relating to the exception.
Return Value	void	
Constraint	Shall be called by TE to log catching a timeout. This event occurs after the catch timeout has been performed.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.59 tliCCreate

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

7.3.4.1.60 tliCStart

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

7.3.4.1.61 tliCRunning

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

7.3.4.1.62 tliCAlive

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

7.3.4.1.63 tliCStop

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

7.3.4.1.64 tliCKill

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

7.3.4.1.65 tliCDoneMismatch

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

7.3.4.1.66 tliCDone

Signature	void tliCDone (in TString am, in TInteger ts, in TString src,in TInteger line, in TriComponentIdType c,in TciNonValueTemplate compTpl)	
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.
	compTpl	The template used to check the done match.
Return Value	void	
Constraint	Shall be called by TE to log the done component operation. This event occurs after the done operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.67 tliCKilledMismatch

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

7.3.4.1.68 tliCKilled

Signature	void tliCKilled (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTpl)	
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.
	compTpl	The template used to check the killed match.
Return Value	void	
Constraint	Shall be called by TE to log the killed component operation. This event occurs after the killed operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.69 tliCTerminated

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

7.3.4.1.70 tliPConnect

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

7.3.4.1.71 tliPDisconnect

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

7.3.4.1.72 tliPMap

Signature	void tliPMap(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	port1	The first port to be mapped.
	port2	The second port to be mapped.
Return Value	void	
Constraint	Shall be called by SA or TE to log the map operation. This event occurs after the map operation.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.73 tliPUnmap

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

7.3.4.1.74 tliPClear

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

7.3.4.1.75 tliPStart

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

7.3.4.1.76 tliPStop

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

7.3.4.1.77 tliPHalt

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

7.3.4.1.78 tliEncode

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

7.3.4.1.79 tliDecode

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

7.3.4.1.80 tliTimeoutDetected

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

7.3.4.1.81 tliTTimeoutMismatch

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

7.3.4.1.82 tliTTimeout

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

7.3.4.1.83 tliTStart

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

7.3.4.1.84 tliTStop

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

7.3.4.1.85 tliTRead

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

7.3.4.1.86 tliTRunning

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

7.3.4.1.87 tliSEnter

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

7.3.4.1.88 tliSLeave

Signature	void tliSLeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars, in Value returnValue, in TString kind)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	name	The name of the scope.
	tciPars	The parameters of the scope.
	returnValue	The return value of the scope.
	kind	The kind of the scope.
Return Value	void	
Constraint	Shall be called by TE to log the leaving of a scope. This event occurs after the scoped has been leaved.	
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 the present document.	

7.3.4.1.89 tliVar

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

7.3.4.1.90 tliModulePar

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

7.3.4.1.91 tliGetVerdict

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

7.3.4.1.92 tliSetVerdict

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

7.3.4.1.93 tliLog

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

7.3.4.1.94 tliAEnter

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

7.3.4.1.95 tliALeave

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

7.3.4.1.96 tliANomatch

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

7.3.4.1.97 tliARepeat

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

7.3.4.1.98 tliADefaults

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

7.3.4.1.99 tliAActivate

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

7.3.4.1.100 tliADeactivate

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

7.3.4.1.101 tliAWait

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

7.3.4.1.102 tliAction

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

7.3.4.1.103 tliMatch

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

7.3.4.1.104 tliMatchMismatch

Signature	void tliMatchMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value expr, in TciValueTemplate tmpl, in TciValueDifferenceList diffs)	
In Parameters	am	An additional message.
	ts	The time when the event is produced.
	src	The source file of the test specification.
	line	The line number where the request is performed.
	c	The component which produces this event.
	expr	The expression to be matched with tmpl
	tmpl	The template to be matched with expr
	diffs	The difference/the mismatch between expr and tmpl
Return Value	void	
Constraint	Shall be called by TE to log that the component unsuccessfully executed a match operation – a mismatch occurred.	
Effect	The TL presents all the information provided in the parameters of this operation to the user, how this is done is not within the scope of the present document.	

7.3.4.1.105 tliInfo

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

8 Java language mapping

8.1 Introduction

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

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

8.2 Names and scopes

8.2.1 Names

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

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

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

The resulting mapping conforms to the standard Java coding conventions.

8.2.2 Scopes

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

Type mapping

Basic type mapping

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

Table 2: Basic type mappings

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

`boolean`

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

`float`

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

`char`

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

`int`

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

`String`

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

`String[]`

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

Universal Char

The IDL **TUniversalChar** type is mapped to a java basic type `int`. The integer uses the canonical form as defined in ISO/IEC 10646-1 [5], clause 6.2.

Structured type mapping

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

8.2.2.1 TciParameterType

TciParameterType is mapped to the following interface:

```
// TCI IDL TciParameterType
package org.etsi.ttcn.tci;
public interface TciParameter {
    public String    getParameterName();
    public void     setParameterName(String name);
    public int      getParameterPassingMode();
    public void     setParameterPassingMode(TciParameterPassingMode mode);
    public Value    getParameter();
    public void     setParameter(Value parameter);
}
```

Methods:

- `getParameterName ()` Returns the parameter name as defined in the TTCN-3 specification.
- `setParameterName (String name)` Sets the name of this `TciParameter` parameter to `name`.
- `getParameterPassingMode ()` Returns the parameter passing mode of this parameter.
- `setParameterPassingMode (TciParameterPassingMode mode)` Sets the parameter mode of this `TriParameter` parameter to `mode`.
- `getParameter ()` Returns the `Value` parameter of this `TciParameter`, or the `null` object if the parameter contains the distinct value `null`.
- `setParameter (Value parameter)` Sets the `Value` parameter of this `TciParameter` to `parameter`. If the distinct value `null` shall be set to indicate that this parameter holds no value, the Java `null` shall be passed as parameter.

8.2.2.2 TciParameterPassingModeType

TciParameterPassingModeType is mapped to the following interface:

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

Constants:

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

8.2.2.3 TciParameterListType

TciParameterListType is mapped to the following interface:

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

Methods:

- `size()` Returns the number of parameters in this list.
- `isEmpty()` Returns `true` if this list contains no parameters.
- `getParameters()` Returns an Enumeration over the parameters in the list. The enumeration provides the parameters in the same order as they appear in the list.
- `get(int index)` Returns the `TciParameter` at the specified position.
- `clear()` Removes all parameters from this `TciParameterList`.
- `add(TciParameter parameter)` Adds parameter to the end of this `TciParameterList`.
- `setParameter(TciParameter[] parameters)` Fills this `TciParameterList` with parameters.

8.2.2.4 TciTypeClassType

TciTypeClassType is mapped to the following interface:

```
// TCI IDL TciTypeClassType
package org.etsi.ttcn.tci;
public interface TciTypeClass {
    public final static int ADDRESS = 0 ;
    public final static int ANYTYPE = 1 ;
    public final static int BITSTRING = 2 ;
    public final static int BOOLEAN = 3 ;
    public final static int CHARSTRING = 5 ;
    public final static int COMPONENT = 6 ;
    public final static int ENUMERATED = 7 ;
    public final static int FLOAT = 8 ;
    public final static int HEXSTRING = 9 ;
    public final static int INTEGER = 10 ;
    public final static int OCTETSTRING = 12 ;
    public final static int RECORD = 13 ;
    public final static int RECORD_OF = 14 ;
    public final static int SET = 15 ;
    public final static int SET_OF = 16 ;
    public final static int UNION = 17 ;
    public final static int UNIVERSAL_CHARSTRING = 19 ;
    public final static int VERDICT = 20 ;
    public final static int ARRAY = 21 ;
}
```

8.2.2.5 TciTestComponentKindType

TciTestComponentKindType is mapped to the following interface:

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

8.2.2.6 TciBehaviourIdType

TciBehaviourIdType is mapped to the following interface:

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

8.2.2.7 TciTestCaseIdType

TciTestCaseIdType is mapped to the following interface:

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

8.2.2.8 TciModuleIdType

TciModuleIdType is mapped to the following interface:

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

8.2.2.9 TciModuleParameterIdType

TciModuleParameterIdType is mapped to the following interface:

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

8.2.2.10 TciModuleParameterListType

TciModuleParameterListType is mapped to the following interface:

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

Methods:

- `size()` Returns the number of module parameters in this list.
- `isEmpty()` Returns `true` if this list contains no module parameters.
- `getModuleParameters()` Returns an `Enumeration` over the module parameters in the list. The enumeration provides the module parameters in the same order as they appear in the list.
- `get(int index)` Returns the `TciModuleParameter` at the specified position.

8.2.2.11 TciModuleParameterType

TciModuleParameterType is mapped to the following interface:

```
// TCI IDL TciModuleParameterType
package org.etsi.ttcn.tci;
public interface TciModuleParameter {
    public String getModuleParameterName();
    public Value getDefaultValue();
}
```

Methods:

- `getModuleParameterName ()` Returns the module parameter name as defined in the TTCN-3 specification.
- `getDefaultValue()` Returns the default Value module parameter of this `TciModuleParameter`, or the `null` object if the module parameter contains the distinct value `null`.

8.2.2.12 TciParameterTypeListType

TciParameterTypeListType is mapped to the following interface:

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

Methods:

- `size()` Returns the number of parameter types in this list.
- `isEmpty()` Returns `true` if this list contains no parameter types.
- `getParameterTypes()` Returns an `Enumeration` over the parameter types in the list. The enumeration provides the parameter types in the same order as they appear in the list.
- `get(int index)` Returns the `TciParameterType` at the specified position.

8.2.2.13 TciParameterTypeType

TciParameterTypeType is mapped to the following interface:

```
// TCI IDL TciParameterTypeType
package org.etsi.ttcn.tci;
public interface TciParameterType {
    public Type    getParameterType() ;
    public int     getParameterPassingMode();
}
```

Methods:

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

8.2.2.14 TciModuleIdListType

TciModuleIdListType is mapped to the following interface:

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

Methods:

- `size()` Returns the number of modules in this list.
- `isEmpty()` Returns `true` if this list contains no modules.

- `tciGetImportedModules()` Returns an Enumeration over the modules in the list. The enumeration provides the modules in the same order as they appear in the list.
- `get(int index)` Returns the `TciModuleId` at the specified position.

8.2.3 Abstract type mapping

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

8.2.3.1 Type

Type is mapped to the following interface:

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

Methods:

- `getDefiningModule()` Returns the module identifier of the module the type has been defined in. If the type represents a TTCN-3 base type the distinct value `null` will be returned.
- `getName()` Returns name of the type as defined in the TTCN-3 module.
- `getTypeClass()` Returns the type class of the respective type. A value of `TciTypeClassType` can have on of the following constants: `ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OCTETSTRING`, `RECORD`, `RECORD_OF`, `ARRAY`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`, `VERDICT`.
- `newInstance()` Returns a freshly created value of the given type. This initial value of the created value is undefined.
- `getTypeEncoding()` Returns the type encoding attribute as defined in the TTCN-3 module.
- `getTypeEncodingVariant()` This operation returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute has been defined the distinct value `null` will be returned.
- `getTypeExtension()` Returns the type extension attribute as defined in the TTCN-3 module.

8.2.4 Abstract value mapping

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

8.2.4.1 Value

Value is mapped to the following interface:

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

Methods:

- `getType()` Returns the type of the specified value.
- `notPresent()` Returns `true` if the specified value is omit, `false` otherwise.
- `getValueEncoding()` This operation returns the value encoding attribute as defined in TTCN-3, if any. If no encoding attribute has been defined the distinct value `null` will be returned.
- `getValueEncodingVariant()` This operation returns the value encoding variant attribute as defined in TTCN-3, if any. If no encoding variant attribute has been defined the distinct value `null` will be returned.

8.2.4.2 IntegerValue

IntegerValue type is mapped to the following interface:

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

Methods:

- `setInteger(int value)` Sets this **IntegerValue** to the int value `value`.
- `getInteger()` Returns the int value represented by this **IntegerValue**.

8.2.4.3 FloatValue

FloatValue type is mapped to the following interface:

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

Methods:

- `setFloat(float value)` Sets this **FloatValue** to the float value `value`.
- `getFloat()` Returns the float value represented by this **FloatValue**.

8.2.4.4 BooleanValue

BooleanValue type is mapped to the following interface:

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

Methods:

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

8.2.4.5 CharstringValue

CharstringValue is mapped to the following interface:

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

Methods:

- `getString()` Returns the string of the TTCN-3 charstring. The textual representation of the empty TTCN-3 charstring is ' ', while its length is zero.
- `setString(String value)` Sets this CharstringValue to value.
- `getChar(int position)` Returns the char value of the TTCN-3 charstring at position. position 0 denotes the first char of the TTCN-3 charstring. Valid values for position are 0 to length - 1.
- `setChar(int position, char value)` Set the char at position to value. Valid values for position are 0 to length - 1.
- `getLength()` Returns the length of this CharstringValue in chars, zero if the value of this CharstringValue is omit.
- `setLength(int len)` Sets the length of this CharstringValue in chars to len.

8.2.4.6 BitstringValue

BitstringValue is mapped to the following interface:

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

Methods:

- `getString()` Returns the textual representation of this `BitstringValue`, as defined in TTCN-3. E.g. the textual representation of 0101 is "0101"B. The textual representation of the empty TTCN-3 bitstring is ""B, while its length is zero.
- `setString(String value)` Sets the value of this `BitstringValue` according to the textual representation as defined by value. E.g. The value of this `BitstringValue` will be 0101 if the textual representation in value is "0101"B.
- `getBit(int position)` Returns the value (0 | 1) at position of this TTCN-3 bitstring. position 0 denotes the first bit of the TTCN-3 bitstring. Valid values for position are 0 to length - 1.
- `setBit(int position, int value)` Set the bit at position to value (0 | 1). position 0 denotes the first bit in this `BitstringValue`. Valid values for position are 0 to length - 1.
- `getLength()` Returns the length of this `BitstringValue` in bits, zero if the value of this `BitstringValue` is omit.
- `setLength(int len)` Sets the length of this `BitstringValue` in bits to len.

8.2.4.7 OctetstringValue

OctetstringValue is mapped to the following interface:

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

Methods:

- `getString()` Returns the textual representation of this `OctetstringValue`, as defined in TTCN-3. E.g. the textual representation of 0xCAFFEE is "CAFFEE"O. The textual representation of the empty TTCN-3 octetstring is ""O, while its length is zero.
- `setString(String value)` Sets the value of this `OctetstringValue` according to the textual representation as defined by value. E.g. the value of this `OctetstringValue` will be 0xCAFFEE if the textual representation in value is "CAFFEE"O.
- `getOctet(int position)` Returns the value (0..255) at position of this TTCN-3 octetstring. position 0 denotes the first octet of the TTCN-3 octetstring. Valid values for position are 0 to length - 1.
- `setOctet(int position, int value)` Set the octet at position to value (0..255). position 0 denotes the first octet in the octetstring. Valid values for position are 0 to length - 1.
- `getLength()` Returns the length of this `OctetstringValue` in octets, zero if the value of this `OctetstringValue` is omit.

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

8.2.4.8 UniversalCharstringValue

UniversalCharstringValue is mapped to the following interface:

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

Methods:

- `getString()` Returns the textual representation of this `UniversalCharstringValue`, as defined in TTCN-3.
- `setString(String value)` Sets the value of this `UniversalCharstringValue` according to the textual representation as defined by `value`.
- `getChar(int position)` Returns the `UniversalChar` value of the TTCN-3 universal charstring at position. `position 0` denotes the first `UniversalChar` of the TTCN-3 universal charstring. Valid values for `position` are `0` to `length - 1`.
- `setChar(int position, char value)` Set the `UniversalChar` at position to `value`. Valid values for `position` are `0` to `length - 1`.
- `getLength()` Returns the length of this `UniversalCharstringValue` in `UniversalChars`, zero if the value of this `UniversalCharstringValue` is omit.
- `setLength(int len)` Sets the length of this `UniversalCharstringValue` in `UniversalChars` to `len`.

8.2.4.9 HexstringValue

HexstringValue is mapped to the following interface:

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

Methods:

- `getString()` Returns the textual representation of this `HexstringValue`, as defined in TTCN-3. E.g. the textual representation of `0xAFFEE` is `"AFFEE" H`. The textual representation of the empty TTCN-3 `hexstring` is `" H`, while its length is zero.
- `setString(String value)` Sets the value of this `HexstringValue` according to the textual representation as defined by `value`. E.g. the value of this `HexstringValue` will be `0xAFFEE` if the textual representation in `value` is `"AFFEE" H`.
- `getHex(int position)` Returns the value (0...15) at `position` of this TTCN-3 `hexstring`. `position 0` denotes the first hex digits of the TTCN-3 `hexstring`. Valid values for `position` are 0 to `length - 1`.
- `setHex(int position, int value)` Set the hex digit at `position` to `value` (0...16). `position 0` denotes the first octet in the `hexstring`. Valid values for `position` are 0 to `length - 1`.
- `getLength()` Returns the length of this `HexstringValue` in octets, zero if the value of this `HexstringValue` is `omit`.
- `setLength(int len)` Sets the length of this `HexstringValue` in hex digits to `len`.

8.2.4.10 RecordValue

RecordValue is mapped to the following interface:

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

```

Methods:

- `getField(String fieldName)` Returns the value of the field named `fieldName`. The return value is the common abstract base type `Value`, as a record field can have any type defined in TTCN-3. If the field cannot be obtained from the record the distinct value `null` will be returned.
- `setField(String fieldName, Value value)` Set the field named `fieldName` of the record to `value`. No assumption shall be made on how a field is stored in a record. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore it should be assumed that subsequent modifications of `value` will not be considered in the record.
- `getFieldNames()` Returns an array of `String` of field names, the empty sequence, if the record has no fields.
- `setFieldOmitted(String fieldName)` Set the field named `fieldName` of the record to `omit`.

8.2.4.11 RecordOfValue

RecordOfValue is mapped to the following interface:

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

Methods:

- `getField(String fieldName)` Returns the value of the record of at position if position is between zero and length -1, the distinct value null otherwise. The return value is the common abstract base type Value, as a record of can have fields of any type defined in TTCN-3.
- `setField(int position, Value value)` Sets the field at position to value. If position is greater than (length -1) the record of will be extended to have the length (position + 1). The record of elements between the original position at length and position - 1 will be set to OMIT. No assumption shall be made on how a field is stored in a record of. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore it should be assumed that subsequent modifications of value will not be considered in the record of.
- `appendField(Value value)` Appends the value at the end of the record of, i.e. at position length. No assumption shall be made on how a field is stored in a record of. An internal implementation might choose to store a reference to this value or to copy the value. It is safe to assume that the value will be copied. Therefore it should be assumed that subsequent modifications of value will not be considered in the record of.
- `getElementType()` This operation will return the Type of the elements of this record of.
- `getLength()` Returns the actual length of the record of value, zero if the record of value is OMIT.
- `setLength(int len)` Set 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 or equal than the original length this operation will be ignored.
- `getOffset()` Returns the lowest possible index. For a record of or set of value this is always 0. For an array value, this is the lower index bound used in the type definition.

8.2.4.12 UnionValue

UnionValue is mapped to the following interface:

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

Methods:

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

8.2.4.13 EnumeratedValue

EnumeratedValue is mapped to the following interface:

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

Methods:

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

8.2.4.14 VerdictValue

VerdictValue is mapped to the following interface:

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

```

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

```

Methods:

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

8.2.4.15 AddressValue

AddressValue is mapped to the following interface:

```

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

```

Methods:

- `getAddress()` Returns the value represented by this `AddressValue`.
- `setAddress(Value value)` Sets this `AddressValue` to the value `value`.

8.2.5 Abstract logging types mapping

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

8.2.5.1 TciValueTemplate

TciValueTemplate is mapped to the following interface:

```

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

```

Methods:

- `isOmit ()` Returns `true` if the template is `omit`, `false` otherwise.
- `isAny ()` Returns `true` if the template is `any`, `false` otherwise.
- `isAnyOrOmit ()` Returns `true` if the template is `anyoromit`, `false` otherwise.
- `getTemplateDef ()` This operation returns the template definition.

8.2.5.2 TciNonValueTemplate

TciNonValueTemplate is mapped to the following interface:

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

Methods:

- `isAny()` Returns `true` if the template is any, `false` otherwise.
- `isAll()` Returns `true` if the template is all, `false` otherwise.
- `getTemplateDef()` This operation returns the template definition.

8.2.5.3 TciValueList

TciValueList is mapped to the following interface:

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

Methods:

- `size()` Returns the number of values in this list.
- `isEmpty()` Returns `true` if this list contains no values.
- `get(int index)` Returns the `Value` at the specified position.

8.2.5.4 TciValueDifference

TciValueDifference is mapped to the following interface:

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

Methods:

- `getValue()` Returns the value of this `TciValueDifference`.
- `getTciValueTemplate ()` Returns the template of this `TciValueDifference`.
- `getDescription()` Returns the description of the mismatch.

8.2.5.5 TciValueDifferenceList

TciValueDifferenceList is mapped to the following interface:

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

Methods:

- `size()` Returns the number of differences in this list.
- `isEmpty()` Returns `true` if this list contains no differences.
- `get(int index)` Returns the `TciValueDifference` at the specified position.

8.3 Constants

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

The following constants can be used to determine the parameter passing mode of TTCN-3 parameters (see also clause 8.2.2.3).

- `org.etsi.ttcn.tci.TciParameterPassingMode.TCI_IN;`
- `org.etsi.ttcn.tci.TciParameterPassingMode.TCI_INOUT;`
- `org.etsi.ttcn.tri.TciParameterPassingMode.TCI_OUT.`

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

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

- `org.etsi.ttcn.tci.TciTypeClass.ADDRESS;`
- `org.etsi.ttcn.tci.TciTypeClass.ANYTYPE;`
- `org.etsi.ttcn.tci.TciTypeClass.BITSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.BOOLEAN;`
- `org.etsi.ttcn.tci.TciTypeClass.CHARSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.COMPONENT;`
- `org.etsi.ttcn.tci.TciTypeClass.ENUMERATED;`
- `org.etsi.ttcn.tci.TciTypeClass.FLOAT;`
- `org.etsi.ttcn.tci.TciTypeClass.HEXSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.INTEGER;`
- `org.etsi.ttcn.tci.TciTypeClass.OCTETSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.RECORD;`
- `org.etsi.ttcn.tci.TciTypeClass.RECORD_OF;`
- `org.etsi.ttcn.tci.TciTypeClass.SET;`
- `org.etsi.ttcn.tci.TciTypeClass.SET_OF;`

- `org.etsi.ttcn.tci.TciTypeClass.ARRAY;`
- `org.etsi.ttcn.tci.TciTypeClass.UNION;`
- `org.etsi.ttcn.tci.TciTypeClass.UNIVERSAL_CHARSTRING;`
- `org.etsi.ttcn.tci.TciTypeClass.VERDICT.`

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

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

8.4 Mapping of interfaces

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

Table 3: Sub interfaces

Calling/Called	TE	TM	CD	CH	SA	PA	TL
TE	-	TCI-TM Provided	TCI-CD Provided	TCI-CH Provided	See ES 201 873-5 [3]	See ES 201 873-5 [3]	TCI-TL Provided
TM	TCI-TM Required	-	-	-	-	-	TCI-TL Provided
CD	TCI-CD Required	-	-	-	-	-	TCI-TL Provided
CH	TCI-CH Required	-	-	-	-	-	TCI-TL Provided
SA	See ES 201 873-5 [3]	-	-	-	-	-	TCI-TL Provided
PA	See ES 201 873-5 [3]	-	-	-	-	-	TCI-TL Provided
TL	-	-	-	-	-	-	-

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

8.4.1 The TCI-TM interface

8.4.1.1 TCI-TM provided

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

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

```

    public void    tciError (String message);
}

```

8.4.1.2 TCI-TM required

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

```

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

```

8.4.2 The TCI-CD interface

8.4.2.1 TCI-CD provided

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

```

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

```

8.4.2.2 TCI-CD required

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

```

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

```

8.4.3 The TCI-CH interface

8.4.3.1 TCI-CH provided

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

```

// TciCHProvided
// TE -> CH
package org.etsi.ttcn.tci;
public interface TciCHProvided {

```

```

public void tciSendConnected (TriPortId sender, TriComponentId receiver, Value sendMessage) ;
public void tciSendConnectedBC (TriPortId sender, Value sendMessage) ;
public void tciSendConnectedMC (TriPortId sender, TriComponentIdList receivers,
Value sendMessage) ;

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

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

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

public TriComponentId tciCreateTestComponentReq(int kind, Type componentType, String name) ;
public void tciStartTestComponentReq(TriComponentId comp,
TciBehaviourId behaviour,
TciParameterList parameterList) ;

public void tciStopTestComponentReq(TriComponentId comp) ;
public void tciConnectReq(TriPortId fromPort, TriPortId toPort) ;
public void tciDisconnectReq(TriPortId fromPort, TriPortId toPort) ;
public void tciTestComponentTerminatedReq(TriComponentId comp, VerdictValue verdict) ;
public boolean tciTestComponentRunningReq(TriComponentId comp) ;
public TriComponentId tciGetMTCReq() ;
public void tciMapReq(TriPortId fromPort, TriPortId toPort);
public void tciUnmapReq(TriPortId fromPort, TriPortId toPort);
public void tciExecuteTestCaseReq(TriComponentId testComponentId,
TriPortIdList tsiPortList);

public void tciResetReq() ;
public boolean tciTestComponentDoneReq(TriComponentId testComponentId) ;
public void tciKillTestComponentReq(TriComponentId component)
public boolean tciTestComponentAliveReq (TriComponentId component)
public boolean tciTestComponentKilledReq (TriComponentId component)

```

```
}

```

8.4.3.2 TCI-CH required

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

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

    public void    tciEnqueueCallConnected(TriPortId sender,
                                         TriComponentId receiver,
                                         TriSignatureId signature,
                                         TciParameterList parameterList) ;

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

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

    public TriComponentId    tciCreateTestComponent(int kind, Type componentType, String name) ;
    public void    tciStartTestComponent(TriComponentId comp,
                                         TciBehaviourId behaviour,
                                         TciParameterList parameterList) ;

    public void    tciStopTestComponent(TriComponentId comp) ;
    public void    tciConnect(TriPortId fromPort, TriPortId toPort) ;
    public void    tciDisconnect(TriPortId fromPort, TriPortId toPort) ;
    public void    tciTestComponentTerminated(TriComponentId comp, VerdictValue verdict);
    public boolean tciTestComponentRunning(TriComponentId comp);
    public boolean tciTestComponentDone(TriComponentId comp);
    public TriComponentId    tciGetMTC ();
    public void    tciExecuteTestCase (TciTestCaseId TestCaseId, TriPortIdList tsiPortList);
    public void    tciReset ();
    public void    tciMap (TriPortId fromPort, TriPortId toPort);
    public void    tciUnmap (TriPortId fromPort, TriPortId toPort);
    public void    tciKillTestComponent(TriComponentId component)
    public boolean tciTestComponentAlive (TriComponentId component)
    public boolean tciTestComponentKilled (TriComponentId component)
}

```

8.4.4 The TCI-TL interface

8.4.4.1 TCI-TL provided

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

```
// TCI-TL
// TE, TM,CH,CD, SA,PA -> TL
package org.etsi.ttcn.tci;
public interface TciTLProvided {
    public void tliTcExecute(String am, int ts, String src, int line, TriComponentId c,
        TciTestCaseId tcId, TriParameterList triPars, TriTimerDuration dur);
    public void tliTcStart(String am, int ts, String src, int line, TriComponentId c,
        TciTestCaseId tcId, TriParameterList tciPars, TriTimerDuration dur);
    public void tliTcStop(String am, int ts, String src, int line, TriComponentId c);
    public void tliTcStarted(String am, int ts, String src, int line, TriComponentId c,
        TciTestCaseId tcId, TriParameterList tciPars, TriTimerDuration dur);
    public void tliTcTerminated(String am, int ts, String src, int line, TriComponentId c,
        TciTestCaseId tcId, TriParameterList tciPars, VerdictValue verdict);
    public void tliCtrlStart(String am, int ts, String src, int line, TriComponentId c);
    public void tliCtrlStop(String am, int ts, String src, int line, TriComponentId c);
    public void tliCtrlTerminated(String am, int ts, String src, int line, TriComponentId c);
    public void tliMSend_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, Value msgValue, Value addrValue,
        TciStatus encoderFailure, TriMessage msg, TriAddress address,
        TriStatus transmissionFailure);
    public void tliMSend_m_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, Value msgValue,
        TciStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_m_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, Value msgValue, TciValueList addrValues,
        TciStatus encoderFailure, TriMessage msg, TriAddressList addresses,
        TriStatus transmissionFailure);
    public void tliMSend_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, Value msgValue, TriStatus transmissionFailure);
    public void tliMSend_c_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortIdList to, Value msgValue, TriStatus transmissionFailure);
    public void tliMSend_c_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortIdList to, Value msgValue, TriStatus transmissionFailure);
    public void tliMDetected_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId from, TriMessage msg, TriAddress address);
    public void tliMDetected_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId from, Value msgValue );
    public void tliMMismatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs,
        Value addrValue, TciValueTemplate addressTmpl);
    public void tliMMismatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs,
        TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliMReceive_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, Value msgValue, TciValueTemplate msgTmpl,
        Value addrValue, TciValueTemplate addressTmpl);
    public void tliMReceive_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, Value msgValue, TciValueTemplate msgTmpl,
        TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrCall_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
        Value addrValue, TciStatus encoderFailure, TriParameterList triPars,
        TriAddress address, TriStatus transmissionFailure);
    public void tliPrCall_m_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
        TciStatus encoderFailure, TriParameterList triPars,
        TriStatus transmissionFailure);
    public void tliPrCall_m_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
        TciValueList addrValues, TciStatus encoderFailure, TriParameterList triPars,
        TriAddressList addresses, TriStatus transmissionFailure);
    public void tliPrCall_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortId to, TriSignatureId signature, TciParameterList tciPars,
        TriStatus transmissionFailure);
    public void tliPrCall_c_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
        TriStatus transmissionFailure);
    public void tliPrCall_c_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
        TriStatus transmissionFailure);
}
```

```

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

```



```

public void tliPrRaise_m_BC(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortId to,
    TriSignatureId signature, TciParameterList tciPars, Value excValue,
    TciStatus encoderFailure, TriException exc, TriStatus transmissionFailure);
public void tliPrRaise_m_MC(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortId to,
    TriSignatureId signature, TciParameterList tciPars, Value excValue,
    TciValueList addrValues, TciStatus encoderFailure, TriException exc,
    TriAddressList addresses, TriStatus transmissionFailure);
public void tliPrRaise_c(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortId to, TriSignatureId signature,
    TciParameterList tciPars, Value excValue, TriStatus transmissionFailure);
public void tliPrRaise_c_BC(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
    Value excValue, TriStatus transmissionFailure);
public void tliPrRaise_c_MC(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortIdList to, TriSignatureId signature, TciParameterList tciPars,
    Value excValue, TriStatus transmissionFailure);
public void tliPrCatchDetected_m(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortId from, TriSignatureId signature,
    TriException exc, TriAddress address);
public void tliPrCatchDetected_c(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriPortId from, TriSignatureId signature,
    Value excValue);
public void tliPrCatchMismatch_m(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature,
    Value excValue, TciValueTemplate excTpl, TciValueDifferenceList diffs,
    Value addrValue, TciValueTemplate addressTpl);
public void tliPrCatchMismatch_c(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature,
    Value excValue, TciValueTemplate excTpl, TciValueDifferenceList diffs,
    TriComponentId from, TciNonValueTemplate fromTpl);
public void tliPrCatch_m(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature,
    Value excValue, TciValueTemplate excTpl,
    Value addrValue, TciValueTemplate addressTpl);
public void tliPrCatch_c(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature,
    Value excValue, TciValueTemplate excTpl,
    TriComponentId from, TciNonValueTemplate fromTpl);
public void tliPrCatchTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature);
public void tliPrCatchTimeout(String am, int ts, String src, int line, TriComponentId c,
    TriPortId at, TriSignatureId signature);
public void tliCCreate(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, String name, Boolean alive);
public void tliCStart(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, TciBehaviourId name, TciParameterList tciPars);
public void tliCRunning(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, ComponentStatus status);
public void tliCAlive(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, ComponentStatus status);
public void tliCStop(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp);
public void tliCKill(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp);
public void tliCDoneMismatch(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, TciNonValueTemplate compTpl);
public void tliCDone(String am, int ts, String src, int line, TriComponentId c,
    TciNonValueTemplate compTpl);
public void tliCKilledMismatch(String am, int ts, String src, int line, TriComponentId c,
    TriComponentId comp, TciNonValueTemplate compTpl);
public void tliCKilled(String am, int ts, String src, int line, TriComponentId c,
    TciNonValueTemplate compTpl);
public void tliCTerminated(String am, int ts, String src, int line, TriComponentId c,
    VerdictValue verdict);
public void tliPConnect(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port1, TriPortId port2);
public void tliPDisconnect(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port1, TriPortId port2);
public void tliPMap(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port1, TriPortId port2);
public void tliPUnmap(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port1, TriPortId port2);
public void tliPClear(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port);
public void tliPStart(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port);

```

```

public void tliPStop(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port);
public void tliPHalt(String am, int ts, String src, int line, TriComponentId c,
    TriPortId port);
public void tliEncode(String am, int ts, String src, int line, TriComponentId c,
    Value val, TciStatus encoderFailure, TriMessage msg, String codec);
public void tliDecode(String am, int ts, String src, int line, TriComponentId c,
    TriMessage msg, TciStatus decoderFailure, Value val, String codec);
public void tliTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer);
public void tliTimeoutMismatch(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, TciNonValueTemplate timerTpl);
public void tliTimeout(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, TciNonValueTemplate timerTpl);
public void tliStart(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, TriTimerDuration dur);
public void tliStop(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, in TriTimerDuration dur);
public void tliRead(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, TriTimerDuration elapsed);
public void tliRunning(String am, int ts, String src, int line, TriComponentId c,
    TriTimerId timer, TimerStatus status);
public void tliEnter(String am, int ts, String src, int line, TriComponentId c,
    QualifiedName name, TciParameterList tciPars, String kind);
public void tliLeave(String am, int ts, String src, int line, TriComponentId c,
    QualifiedName name, TciParameterList tciPars, Value returnValue, String kind);
public void tliVar(String am, int ts, String src, int line, TriComponentId c,
    QualifiedName name, Value varValue);
public void tliModulePar(String am, int ts, String src, int line, TriComponentId c,
    QualifiedName name, Value parValue);
public void tliGetVerdict(String am, int ts, String src, int line, TriComponentId c,
    VerdictValue verdict);
public void tliSetVerdict(String am, int ts, String src, int line, TriComponentId c,
    VerdictValue verdict, String reason);
public void tliLog(String am, int ts, String src, int line, TriComponentId c,
    TString log);
public void tliAEnter(String am, int ts, String src, int line, TriComponentId c);
public void tliALeave(String am, int ts, String src, int line, TriComponentId c);
public void tliADefaults(String am, int ts, String src, int line, TriComponentId c);
public void tliAActivate(String am, int ts, String src, int line, TriComponentId c,
    QualifiedName name, TciParameterList tciPars, Value ref);
public void tliADeactivate(String am, int ts, String src, int line, TriComponentId c,
    Value ref);
public void tliANomatch(String am, int ts, String src, int line, TriComponentId c);
public void tliARepeat(String am, int ts, String src, int line, TriComponentId c);
public void tliAwait(String am, int ts, String src, int line, TriComponentId c);
public void tliAction(String am, int ts, String src, int line, TriComponentId c, String action);
public void tliMatch(String am, int ts, String src, int line, TriComponentId c, Value expr,
    TciValueTemplate tpl);
public void tliMatchMismatch(String am, int ts, String src, int line, TriComponentId c,
    Value expr, TciValueTemplate tpl, TciValueDifferenceList diffs);
public void tliInfo (String am, int ts, String src, int line, TriComponent c,
    int level, String info)
}

```

8.5 Optional parameters

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

8.6 TCI initialization

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

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

8.7 Error handling

All operations called from the TM, CH or CD that return have succeeded. If an erroneous situation has been identified by the TE a test case error will be communicated to the user using the procedures as defined in clause 7.3.1.2.6 (`tciError`). If an operation called by the TE in the TM, CH, CD, or TL produces an error, this erroneous situation should be communicated to the TE using the procedures as defined in clause 7.3.2.1.12 (`tciErrorReq`).

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

9 ANSI C language mapping

9.1 Introduction

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

9.2 Value interfaces

TCI IDL Interface	ANSI C representation	Notes and comments
Type		
<code>TciModuleIdType getDefiningModule()</code>	<code>TciModuleIdType tciGetDefiningModule(Type inst)</code>	
<code>TString getName()</code>	<code>String tciGetName(Type inst)</code>	String type reused from IDL (OMG recommendation)
<code>TciTypeClassType getTypeClass()</code>	<code>TciTypeClassType tciGetTypeClass (Type inst)</code>	
<code>Value newInstance()</code>	<code>Value tciNewInstance(Type inst)</code>	
<code>TString getTypeEncoding()</code>	<code>String tciGetTypeEncoding(Type inst)</code>	
<code>TStringSeq getTypeExtension()</code>	<code>String* getTypeExtension(Type inst)</code>	
<code>TString getTypeEncodingVariant()</code>	<code>String tciGetTypeEncodingVariant (Type inst)</code>	
Value		
<code>TString getValueEncoding()</code>	<code>String tciGetValueEncoding (Value inst)</code>	
<code>TString getValueEncodingVariant()</code>	<code>String tciGetValueEncodingVariant (Value inst)</code>	
<code>Type getType()</code>	<code>Type tciGetType (Value inst)</code>	
<code>TBoolean notPresent()</code>	<code>Boolean tciNotPresent (Value inst)</code>	Boolean type reused from IDL (OMG recommendation)
	<code>void tciSetNull (Value inst)</code>	For optional parameters of operations, see clause 9.7.
	<code>Boolean tciIsNull (Value inst)</code>	For optional parameters of operations, see clause 9.7. Boolean type reused from IDL (OMG recommendation)
IntegerValue		
<code>TInteger getInt()</code>	<code>String tciGetIntAbs (Value inst)</code>	Returns the (10-base) integer absolute value as an ASCII string.

TCI IDL Interface	ANSI C representation	Notes and comments
	unsigned long int tciGetIntNumberOfDigits (Value inst)	Returns the number of digits in an integer value.
	Boolean tciGetIntSign(Value inst)	Returns true if the number is positive, false otherwise.
	char tciGetIntDigit (Value inst, unsigned long int position)	Returns the value of the digit at position 'position', where position 0 is the least significant digit.
void setInt(in TInteger value)	void tciSetIntAbs(Value inst, String value)	Sets the (10-base) absolute value of the integer via an ASCII string. The first digit must not be 0 unless the value is 0.
	void tciSetIntNumberOfDigits (Value inst, unsigned long int dig_num)	Sets the number of digits in an integer value.
	void tciSetIntSign (Value inst, Boolean sign)	Sets the sign to + (true) or - (false).
	void tciSetIntDigit (Value inst, unsigned long int position, char digit)	Sets the value of the digit at position 'position', where position 0 is the least significant digit.
FloatValue		
TFloat getFloat()	double tciGetFloatValue(Value inst)	
void setFloat(in TFloat value)	void tciSetFloatValue(Value inst, double value)	
BooleanValue		
TBoolean getBoolean()	Boolean tciGetBooleanValue(Value inst)	
void setBoolean (in TBoolean value)	void tciSetBooleanValue (Value inst, Boolean value)	
CharstringValue		
TString getString()	TciCharStringValue tciGetCStringValue(Value inst)	
void setString(in TString value)	void tciSetCStringValue (Value inst, TciCharStringValue value)	
TChar getChar (in TInteger position)	char tciGetCStringCharValue (Value inst, long int position)	
void setChar (in TInteger position, in char value)	void tciSetCStringCharValue (Value inst, long int position, char value)	
TInteger getLength()	unsigned long int tciGetCStringLength (Value inst)	
void setLength(in TInteger len)	void tciSetCStringLength (Value inst, unsigned long int len)	
UniversalCharstringValue		
TString getString()	TciUCStringValue tciGetUCStringValue(Value inst)	
void setString(in TString value)	void tciSetUCStringValue (Value inst, TciUCStringValue value)	
TUniversalChar getChar (in TInteger position)	void tciGetUCStringCharValue (Value inst, unsigned long int position, TciUCValue result)	
void setChar (in TInteger position, in TUniversalChar value)	void tciSetUCStringCharValue (Value inst, unsigned long int position, TciUCValue value)	
TInteger getLength()	unsigned long int tciGetUCStringLength(Value inst)	

TCI IDL Interface	ANSI C representation	Notes and comments
void setLength(in TInteger len)	void tciSetUCStringLength (Value inst, unsigned long int len)	
BitstringValue		
TString getString()	String tciGetBStringValue(Value inst)	
void setString(in TString value)	void tciSetBStringValue (Value inst, String value)	
TChar getBit (in integer position)	int tciGetBStringBitValue (Value inst, long int position)	
void setBit (in TInteger position, in TInteger value)	void tciSetBStringBitValue (Value inst, unsigned long int position, int value)	
TInteger getLength()	unsigned long int tciGetBStringLength(Value inst)	
void setLength(in TInteger len)	void tciSetBStringLength (Value inst, long int len)	
HexstringValue		
TString getString()	String tciGetHStringValue(Value inst)	
void setString(in TString value)	void tciSetHStringValue (Value inst, String value)	
TChar getHex (in TInteger position)	int tciGetHStringHexValue (Value inst, unsigned long int position)	
void setBit (in TInteger position, in TInteger value)	void tciSetHStringHexValue (Value inst, unsigned long int position, int value)	
TInteger getLength()	long int tciGetHStringLength(Value inst)	
void setLength(in TInteger len)	void tciSetHStringLength (Value inst, unsigned long int len)	
OctetstringValue		
TString getString()	String tciGetOStringValue(Value inst)	
void setString(in TString value)	void tciSetOStringValue (Value inst, String value)	
TChar getOctet(in TInteger position)	int tciGetOStringOctetValue (Value inst, unsigned long int position)	
void setOctet (in TInteger position, in TInteger value)	void tciSetOStringOctetValue (Value inst, unsigned long int position, int value)	
TInteger getLength()	unsigned long int tciGetOStringLength(Value inst)	
void setLength(in TInteger len)	void tciSetOStringLength (Value inst, unsigned long int len)	
RecordValue		
Value getField(in TString fieldName)	Value tciGetRecFieldValue (Value inst, String fieldName)	
void setField (in TString fieldName, in Value value)	void tciSetRecFieldValue (Value inst, String fieldName, Value value)	
TString[] getFieldNames()	char** tciGetRecFieldNames(Value inst)	Returns a NULL-terminated array of the field names.
void setFieldOmitted (in TString fieldName)	void setFieldOmitted (Value inst, String fieldName)	
RecordOfValue		
Value getField(in TInteger position)	Value tciGetRecOfFieldValue (Value inst, unsigned long int position)	
void setField (in TInteger position, in Value value)	void tciSetRecOfFieldValue (Value inst, unsigned long int position, Value value)	
void appendField(in Value value)	void tciAppendRecOfFieldValue (Value inst, Value value)	

TCI IDL Interface	ANSI C representation	Notes and comments
Type getElementType()	Type tciGetRecOfElementType(Value inst)	
TInteger getLength()	unsigned long int tciGetRecOfLength(Value inst)	
void setLength(in TInteger len)	void tciSetRecOfLength (Value inst, unsigned long int len)	
TInteger getOffset()	unsigned long int tciGetOffset(Value inst)	
UnionValue		
Value getVariant (in TString variantName)	Value tciGetUnionVariant (Value inst, String variantName)	
void setVariant (in TString variantName, in Value value)	void tciSetUnionVariant (Value inst, String variantName, Value value)	
TString getPresentVariantName()	String tciGetUnionPresentVariantName (Value inst)	
TString[] getVariantNames()	char** tciGetUnionVariantNames(Value inst)	Returns a NULL-terminated array of the field names.
EnumeratedValue		
TString getEnum()	String tciGetEnumValue(Value inst)	
void setEnum(in TString enumValue)	void tciSetEnumValue (Value inst, String enumValue)	
VerdictValue		
TInteger getVerdict()	int tciGetVerdictValue(Value inst)	
void setVerdict(in TInteger verdict)	void tciSetVerdictValue(Value inst, int verdict)	
AddressValue		
Value getAddress()	Value tciGetAddressValue(Value inst)	
void setAddress(in Value value)	void tciSetAddressValue(Value inst, Value value)	

9.3 Logging interface

TCI IDL Interface	ANSI C representation	Notes and comments
TciValueTemplate		
TBoolean isOmit()	Boolean tciIsOmit(TciValueTemplate inst)	Boolean type reused from IDL (OMG recommendation)
TBoolean isAny()	Boolean tciIsAny(TciValueTemplate inst)	Boolean type reused from IDL (OMG recommendation)
TBoolean isAnyOrOmit()	Boolean tciIsAnyOrOmit(TciValueTemplate inst)	Boolean type reused from IDL (OMG recommendation)
TString getTemplateDef()	String tciGetTemplateDef(TciValueTemplate inst)	String type reused from IDL (OMG recommendation)
TciNonValueTemplate		
TBoolean isAny()	Boolean tciIsAnyNonValue (TciNonValueTemplate inst)	Boolean type reused from IDL (OMG recommendation)
TBoolean isAll()	Boolean tciIsAllNonValue (TciNonValueTemplate inst)	Boolean type reused from IDL (OMG recommendation)
TString getTemplateDef()	String tciGetTemplateDefNonValue (TciNonValueTemplate inst)	String type reused from IDL (OMG recommendation)
TciValueList		
TInteger size()	int size(TciValueList inst)	
TBoolean isEmpty()	Boolean isEmpty(TciValueList inst)	Boolean type reused from IDL (OMG recommendation)
Value get(TInteger index)	Value get(TciValueList inst, int index)	

TCI IDL Interface	ANSI C representation	Notes and comments
TciValueDifference		
Value getValue()	Value getValue(TciValueDifference inst)	
TciValueTemplate getTciValueTemplate()	TciValueTemplate getTciValueTemplate(TciValueDifference inst)	
TString getDescription()	String getDescription(TciValueDifference inst)	String type reused from IDL (OMG recommendation)
TciValueDifferenceList		
TInteger size()	int size(TciValueDifferenceList inst)	
TBoolean isEmpty()	Boolean isEmpty(TciValueDifferenceList inst)	Boolean type reused from IDL (OMG recommendation)
TciValueDifference get(TInteger index)	TciValueDifference get(TciValueDifferenceList inst, int index)	

9.4 Operation interfaces

9.4.1 The TCI-TM interface

9.4.1.1 TCI-TM provided

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

```
void tciTestCaseStarted
    (TciTestCaseIdType testCaseId, TciParameterListType parameterList, double timer)
void tciTestCaseTerminated (VerdictValue verdict, TciParameterListType parameterlist)
void tciControlTerminated()
Value tciGetModulePar (TciModuleParameterIdType parameterId)
void tciLog(String message)
void tciError(String message)
```

9.4.1.2 TCI-TM required

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

```
void tciRootModule(String moduleId)
TciModuleIdListType tciGetImportedModules()
TciModuleParameterListType tciGetModuleParameters(TciModuleIdType moduleName)
TciTestCaseIdListType tciGetTestCases()
TciParameterTypeListType tciGetTestCaseParameters (TciTestCaseIdType testCaseId)
TriPortIdList tciGetTestCaseTSI (TciTestCaseIdType testCaseId)
void tciStartTestCase (TciTestCaseIdType testCaseId, TciParameterListType parameterlist)
void tciStopTestCase()
TriComponentId tciStartControl()
void tciStopControl()
```

9.4.2 The TCI-CD interface

9.4.2.1 TCI-CD provided

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

```
Value tciDecode (BinaryString message, Type decHypothesis)
BinaryString tciEncode(Value value)
```

NOTE: BinaryString type reused from TRI.

9.4.2.2 TCI-CD required

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

```
Type tciGetTypeForName(String typeName)
Type tciGetIntegerType()
Type tciGetFloatType()
Type tciGetBooleanType()
Type tciGetCharType()
Type tciGetUniversalCharType()
Type tciGetTciCharstringType()
Type tciGetUniversalCharstringType()
Type tciGetHexstringType()
Type tciGetBitstringType()
Type tciGetOctetstringType()
Type tciGetVerdictType()
void tciErrorReq(String message)
```

9.4.3 The TCI-CH interface

9.4.3.1 TCI-CH provided

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

```
void tciSendConnected(TriPortId sender, TriComponentId receiver, Value sendMessage)
void tciSendConnectedBC(TriPortId sender, Value sendMessage)
void tciSendConnectedMC
  (TriPortId sender, TriComponentIdList receivers, Value sendMessage)
void tciCallConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature,
  TciParameterListType parameterList)
void tciCallConnectedBC
  (TriPortId sender, TriSignatureId signature, TciParameterListType parameterList)
void tciCallConnectedMC
  (TriPortId sender, TriComponentIdList receivers, TriSignatureId signature,
  TciParameterListType parameterList)
void tciReplyConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature,
  TciParameterListType parameterList, Value returnValue)
void tciReplyConnectedBC
  (TriPortId sender, TriSignatureId signature, TciParameterListType parameterList,
  Value returnValue)
void tciReplyConnectedMC
  (TriPortId sender, TriComponentIdList receivers, TriSignatureId signature,
  TciParameterListType parameterList, Value returnValue)
void tciRaiseConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature, Value exception)
void tciRaiseConnectedBC
  (TriPortId sender, TriSignatureId signature, Value exception)
void tciRaiseConnectedMC
  (TriPortId sender, TriComponentIdList receivers, TriSignatureId signature, Value exception)
TriComponentId tciCreateTestComponentReq
  (TciTestComponentKindType kind, Type componentType, String name)
void tciStartTestComponentReq
  (TriComponentId component, TciBehaviourIdType behaviour, TciParameterListType parameterList)
void tciStopTestComponentReq (TriComponentId component)
void tciConnectReq(TriPortId fromPort, TriPortId toPort)
void tciDisconnectReq(TriPortId fromPort, TriPortId toPort)
void tciMapReq(TriPortId fromPort, TriPortId toPort)
void tciUnmapReq(TriPortId fromPort, TriPortId toPort)
void tciTestComponentTerminatedReq(TriComponentId component, VerdictValue verdict)
Boolean tciTestComponentRunningReq(TriComponentId component)
Boolean tciTestComponentDoneReq(TriComponentId component)
TriComponentId tciGetMTCReq()
void tciExecuteTestCaseReq(TciTestCaseIdType testCaseId, TriPortIdList tsiPortList)
void tciResetReq()
void tciKillTestComponentReq(TriComponentId component)
Boolean tciTestComponentAliveReq (TriComponentId component)
Boolean tciTestComponentKilledReq (TriComponentId component)
```


9.4.3.2 TCI-CH required

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

```

void          tciEnqueueMsgConnected
  (TriPortId sender, TriComponentId receiver, Value rcvdMessage)
void          tciEnqueueCallConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature,
   TciParameterListType parameterList)
void          tciEnqueueReplyConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature,
   TciParameterListType parameterList, Value returnValue)
void          tciEnqueueRaiseConnected
  (TriPortId sender, TriComponentId receiver, TriSignatureId signature, Value exception)
TriComponentId tciCreateTestComponent
  (TciTestComponentKindType kind, Type componentType, String name)
void          tciStartTestComponent
  (TriComponentId component, TciBehaviourIdType behaviour, TciParameterListType parameterList)
void          tciStopTestComponent(TriComponentId component)
void          tciConnect(TriPortId fromPort, TriPortId toPort)
void          tciDisconnect(TriPortId fromPort, TriPortId toPort)
void          tciMap (TriPortId fromPort, TriPortId toPort)
void          tciUnmap(TriPortId fromPort, TriPortId toPort)
void          tciTestComponentTerminated(TriComponentId component, VerdictValue verdict)
Boolean      tciTestComponentRunning(TriComponentId component)
Boolean      tciTestComponentDone(TriComponentId component)
TriComponentId tciGetMTC()
void          tciExecuteTestCase(TciTestCaseIdType testCaseId, TriPortIdList tsiPortList)
void          tciReset()
void          tciKillTestComponent(TriComponentId component)
Boolean      tciTestComponentAlive(TriComponentId component)
Boolean      tciTestComponentKilled(TriComponentId component)

```

9.4.4 The TCI-TL interface

9.4.4.1 TCI-TL provided

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

```

void tliTcExecute
  (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId,
   TciParameterListType tciPars, TriTimerDuration dur)
void tliTcStart
  (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId,
   TciParameterListType tciPars, TriTimerDuration dur)
void tliTcStop
  (String am, int ts, String src, int line, TriComponentId c)
void tliTcStarted
  (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId,
   TciParameterListType tciPars, TriTimerDuration dur)
void tliTcTerminated
  (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId,
   TciParameterListType tciPars, VerdictValue verdict)

void tliCtrlStart(String am, int ts, String src, int line, TriComponentId c)
void tliCtrlStop(String am, int ts, String src, int line, TriComponentId c)
void tliCtrlTerminated(String am, int ts, String src, int line, TriComponentId c)

void tliMSend_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   Value msgValue, Value addrValue, TciStatus encoderFailure, TriMessage msg,
   TriAddress address, TriStatus transmissionFailure)
void tliMSend_m_BC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   Value msgValue, TciStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure)
void tliMSend_m_MC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   Value msgValue, TciValueList addrValues, TciStatus encoderFailure, TriMessage msg,
   TriAddressList addresses, TriStatus transmissionFailure)
void tliMSend_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   Value msgValue, TriStatus transmissionFailure)
void tliMSend_c_BC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
   Value msgValue, TriStatus transmissionFailure)

```

```

void tliMSend_c_MC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
   Value msgValue, TriStatus transmissionFailure)
void tliMDetected_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
   TriMessage msg, TriAddress address)
void tliMDetected_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
   Value msgValue)
void tliMMismatch_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,
   TciValueTemplate msgTmpl, TciValueDifferenceList diffs, Value addrValue,
   TciValueTemplate addressTmpl)
void tliMMismatch_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,
   TciValueTemplate msgTmpl, TciValueDifferenceList diffs, TriComponentId from,
   TciNonValueTemplate fromTmpl)
void tliMReceive_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,
   TciValueTemplate msgTmpl, Value addrValue, TciValueTemplate addressTmpl)
void tliMReceive_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,
   TciValueTemplate msgTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)

void tliPrCall_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at,
   TriPortId to, TriSignatureId signature,
   TciParameterListType tciPars, Value addrValue, TciStatus encoderFailure,
   TriParameterList triPars, TriAddress address, TriStatus transmissionFailure)
void tliPrCall_m_BC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   TriSignatureId signature, TciParameterListType tciPars, TciStatus encoderFailure,
   TriParameterList triPars, TriStatus transmissionFailure)
void tliPrCall_m_MC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   TriSignatureId signature, TciParameterListType tciPars, TciValueList addrValues,
   TciStatus encoderFailure, TriParameterList triPars, TriAddressList addresses,
   TriStatus transmissionFailure)
void tliPrCall_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at,
   TriPortId to, TriSignatureId signature,
   TciParameterListType tciPars, TriStatus transmissionFailure)
void tliPrCall_c_BC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
   TriSignatureId signature, TciParameterListType tciPars, TriStatus transmissionFailure)
void tliPrCall_c_MC
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
   TriSignatureId signature, TciParameterListType tciPars, TriStatus transmissionFailure)
void tliPrGetCallDetected_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
   TriSignatureId signature, TriParameterList triPars, TriAddress address)
void tliPrGetCallDetected_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
   TriSignatureId signature, TciParameterListType tciPars)
void tliPrGetCallMismatch_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   TciParameterListType tciPars, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
   Value addrValue, TciValueTemplate addressTmpl)
void tliPrGetCallMismatch_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   TciParameterListType tciPars, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
   TriComponentId from, TciNonValueTemplate fromTmpl)
void tliPrGetCall_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   TciParameterListType tciPars, TciValueTemplate parsTmpl, Value addrValue,
   TciValueTemplate addressTmpl)
void tliPrGetCall_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   TciParameterListType tciPars, TciValueTemplate parsTmpl, TriComponentId from,
   TciNonValueTemplate fromTmpl)
void tliPrReply_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
   TriSignatureId signature, TciParameterListType tciPars, Value replValue,
   Value addrValue, TciStatus encoderFailure, TriParameterList triPars,
   TriParameter repl, TriAddress address, TriStatus transmissionFailure)

```

```

void tliPrReply_m_BC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value replValue,
TciStatus encoderFailure, TriParameterList triPars, TriParameter repl,
TriStatus transmissionFailure)
void tliPrReply_m_MC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value replValue,
TciValueList addrValues, TriStatus encoderFailure, TriParameterList triPars,
TriParameter repl, TriAddressList addresses, TciStatus transmissionFailure)
void tliPrReply_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value replValue,
TriStatus transmissionFailure)
void tliPrReply_c_BC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
TriSignatureId signature, TciParameterListType tciPars, Value replValue,
TriStatus transmissionFailure)
void tliPrReply_c_MC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
TriSignatureId signature, TciParameterListType tciPars, Value replValue,
TriStatus transmissionFailure)
void tliPrGetReplyDetected_m
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
TriSignatureId signature, TriParameterList triPars, TriParameter repl, TriAddress address)
void tliPrGetReplyDetected_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
TriSignatureId signature, TciParameterListType tciPars, Value replValue)
void tliPrGetReplyMismatch_m
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
TciParameterListType tciPars, TciValueTemplate parsTpl, Value replValue,
TciValueTemplate replyTpl, TciValueDifferenceList diffs, Value addrValue,
TciValueTemplate addressTpl)
void tliPrGetReplyMismatch_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
TciParameterListType tciPars, TciValueTemplate parsTpl, Value replValue,
TciValueTemplate replyTpl, TciValueDifferenceList diffs, TriComponentId from,
TciNonValueTemplate fromTpl)
void tliPrGetReply_m
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
TciParameterListType tciPars, TciValueTemplate parsTpl, Value replValue,
TciValueTemplate replyTpl, Value addrValue, TciValueTemplate addressTpl)
void tliPrGetReply_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
TciParameterListType tciPars, TciValueTemplate parsTpl, Value replValue,
TciValueTemplate replyTpl, TriComponentId from, TciNonValueTemplate fromTpl)
void tliPrRaise_m
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue, Value addrValue,
TciStatus encoderFailure, TriException exc, TriAddress address, TriStatus transmissionFailure)
void tliPrRaise_m_BC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue,
TciStatus encoderFailure, TriException exc, TriStatus transmissionFailure)
void tliPrRaise_m_MC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue,
TciValueList addrValues, TciStatus encoderFailure, TriException exc,
TriAddressList addresses, TriStatus transmissionFailure)
void tliPrRaise_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue,
TriStatus transmissionFailure)
void tliPrRaise_c_BC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue,
TriStatus transmissionFailure)
void tliPrRaise_c_MC
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortIdList to,
TriSignatureId signature, TciParameterListType tciPars, Value excValue,
TriStatus transmissionFailure)
void tliPrCatchDetected_m
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
TriSignatureId signature, TriException exc, TriAddress address)
void tliPrCatchDetected_c
(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriPortId from,
TriSignatureId signature, Value excValue)

```

```

void tliPrCatchMismatch_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, Value addrValue,
   TciValueTemplate addressTmpl)
void tliPrCatchMismatch_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, TriComponentId from,
   TciNonValueTemplate fromTmpl)
void tliPrCatch_m
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   Value excValue, TciValueTemplate excTmpl, Value addrValue, TciValueTemplate addressTmpl)
void tliPrCatch_c
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,
   Value excValue, TciValueTemplate excTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)
void tliPrCatchTimeoutDetected
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature)
void tliPrCatchTimeout
  (String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature)

void tliCCreate
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp, String name,
   Boolean alive)
void tliCStart
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp,
   TciBehaviourIdType name, TciParameterListType tciPars)
void tliCRunning
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp,
   ComponentStatus status)
void tliCAlive
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp,
   ComponentStatus status)
void tliCStop
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp)
void tliCKill
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp)
void tliCDoneMismatch
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp,
   TciNonValueTemplate compTmpl)
void tliCDone
  (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate compTmpl)
void tliCTerminated
  (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)
void tliCKilledMismatch
  (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp,
   TciNonValueTemplate compTmpl)
void tliCKilled
  (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate compTmpl)

void tliPConnect
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port1, TriPortId port2)
void tliPDisconnect
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port1,
   TriPortId port2)
void tliPMap
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port1, TriPortId port2)
void tliPUnmap
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port1,
   TriPortId port2)
void tliPClear
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port)
void tliPStart
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port)
void tliPStop
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port)
void tliPHalt
  (String am, int ts, String src, int line, TriComponentId c, TriPortId port)

void tliEncode
  (String am, int ts, String src, int line, TriComponentId c, Value val, TciStatus encoderFailure,
   TriMessage msg, String codec)
void tliDecode
  (String am, int ts, String src, int line, TriComponentId c, TriMessage msg,
   TciStatus decoderFailure, Value val, String codec)

void tliTimeoutDetected
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer)

```

```

void tliTimeoutMismatch
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer,
   TciNonValueTemplate timerTpl)
void tliTimeout
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer,
   TciNonValueTemplate timerTpl)
void tliTStart
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, TriTimerDuration dur)
void tliTStop
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, TriTimerDuration dur)
void tliTRead
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer,
   TriTimerDuration elapsed)
void tliTRunning
  (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, TimerStatus status)

void tliSEnter
  (String am, int ts, String src, int line, TriComponentId c, QualifiedName name,
   TciParameterListType tciPars, String kind)
void tliSLeave
  (String am, int ts, String src, int line, TriComponentId c, QualifiedName name,
   TciParameterListType tciPars, Value returnValue, String kind)

void tliVar
  (String am, int ts, String src, int line, TriComponentId c, QualifiedName name, Value varValue)
void tliModulePar
  (String am, int ts, String src, int line, TriComponentId c, QualifiedName name, Value parValue)

void tliGetVerdict(String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)
void tliSetVerdict
  (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict, String reason)

void tliLog(String am, int ts, String src, int line, TriComponentId c, String log)

void tliAEnter(String am, int ts, String src, int line, TriComponentId c)
void tliALeave(String am, int ts, String src, int line, TriComponentId c)
void tliADefaults(String am, int ts, String src, int line, TriComponentId c)
void tliAActivate
  (String am, int ts, String src, int line, TriComponentId c, QualifiedName name,
   TciParameterListType tciPars, Value ref)
void tliADeactivate(String am, int ts, String src, int line, TriComponentId c, Value ref)
void tliANomatch(String am, int ts, String src, int line, TriComponentId c)
void tliARepeat(String am, int ts, String src, int line, TriComponentId c)
void tliAWait(String am, int ts, String src, int line, TriComponentId c)

void tliAction(String am, int ts, String src, int line, TriComponentId c, String action)

void tliMatch
  (String am, int ts, String src, int line, TriComponentId c, Value expr, TciValueTemplate tpl)
void tliMatchMismatch
  (String am, int ts, String src, int line, TriComponentId c, Value expr, TciValueTemplate tpl,
   TciValueDifferenceList diffs);

void tliInfo
  (String am, int ts, String src, int line, TriComponentId c, int level, String info)

```

9.5 Data

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
TciModuleIdType	QualifiedName	
TciModuleParameterType	typedef struct TciModuleParameterType { String parName; Value defaultValue; } TciModuleParameterType;	
TciModuleParameterListType	typedef struct TciModuleParameterListType { long int length; TciModuleParameterType *modParList; } TciModuleParameterListType;	

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
TciParameterType	<pre>typedef struct TciParameterType { String parName; TciParameterPassingModeType parPassMode; Value parValue; } TciParameterType;</pre>	
TciParameterPassingModeType	<pre>typedef enum { TCI_IN_PAR = 0, TCI_INOUT_PAR = 1, TCI_OUT_PAR = 2 } TciParameterPassingModeType;</pre>	
TciParameterListType	<pre>typedef struct TciParameterListType { long int length; TciParameterType *parList; } TciParameterListType;</pre>	length 0 shall be interpreted as "empty list".
TciParameterTypeListType	<pre>typedef struct TciParameterTypeListType { long int length; TciParameterTypeType *parList; } TciParameterTypeListType;</pre>	length 0 shall be interpreted as "empty list".
TciParameterTypeType	<pre>typedef struct TciParameterTypeType { Type parameterType; TciParameterPassingModeType mode; } TciParameterTypeType;</pre>	
TciTestCaseIdListType	<pre>typedef struct TciTestCaseIdListType { long int length; QualifiedName *idList; } TciTestCaseIdListType;</pre>	length 0 shall be interpreted as "empty list".
TciTypeClassType	<pre>typedef enum { TCI_ADDRESS_TYPE, TCI_ANYTYPE_TYPE, TCI_BITSTRING_TYPE, TCI_BOOLEAN_TYPE, TCI_CHAR_TYPE, TCI_CHARSTRING_TYPE, TCI_COMPONENT_TYPE, TCI_ENUMERATED_TYPE, TCI_FLOAT_TYPE, TCI_HEXSTRING_TYPE, TCI_INTEGER_TYPE, TCI_OCTETSTRING_TYPE, TCI_RECORD_TYPE, TCI_RECORD_OF_TYPE, TCI_ARRAY_TYPE, TCI_SET_TYPE, TCI_SET_OF_TYPE, TCI_UNION_TYPE, TCI_UNIVERSAL_CHAR_TYPE, TCI_UNIVERSAL_CHARSTRING_TYPE, TCI_VERDICT_TYPE } TciTypeClassType;</pre>	
TciTestComponentKindType	<pre>typedef enum { TCI_CTRL_COMP, TCI_MTC_COMP, TCI_PTC_COMP, TCI_SYS_COMP, TCI_ALIVE_COMP } TciTestComponentKindType;</pre>	
TciBehaviourIdType	QualifiedName	

TCI IDL ADT	ANSI C representation (Type definition)	Notes and comments
TciValueDifference	<pre>typedef struct TciValueDifference { Value val; TciValueTemplate tmpl; String desc; } TciValueDifference;</pre>	
TciValueDifferenceList	<pre>typedef struct TciValueDifferenceList { long int length; TciValueDifference* diffList; } TciValueDifferenceList;</pre>	length 0 shall be interpreted as "empty list".

9.6 Miscellaneous

TCI concept	ANSI C representation	Notes and comments
Verdict representation		
NONE	const int TCI_VERDICT_NONE = 0	Since the VerdictValue interface is defined in terms of integers, consensus must be established on which value defines which verdict.
PASS	const int TCI_VERDICT_PASS = 1	
INCONC	const int TCI_VERDICT_INCONC = 2	
FAIL	const int TCI_VERDICT_FAIL = 3	
ERROR	const int TCI_VERDICT_ERROR = 4	
CharstringValue representation		
TciCharString	<pre>typedef struct TciCharStringValue { unsigned long int length; char* string; } TciCharStringValue</pre>	
Universal Character[string] representation		
Universal Char	typedef unsigned char TciUCValue[4]	
Universal Charstring	<pre>typedef struct TciUCStringValue { unsigned long int length; TciUCValue *string; } TciUCStringValue;</pre>	

9.7 Optional parameters

Clause 7.4 defines that a reserved value shall be used to indicate the absence of an optional parameter. For the C language mapping an explicit null shall be used. The function `tciSetNull` can be used to set a value to null and `tciIsNull` can be used to check whether a value represents null. `tciIsNull` returns true if the value is null, false otherwise.

For example, if in the `tciReplyConnected` operation the value parameter shall be omitted, then a value `reply` shall be created and set to null; the operation invocation shall be:

```
tciSetNull(reply);
tciReplyConnected (sender, receiver, signature, parameterList, reply).
```

10 C++ language mapping

10.1 Introduction

This clause introduces the TCI C++ language mapping for the definitions given in clause 7.

10.2 Names and scopes

The namespace `ORG_ETSI_TTCN3_TCI` has been defined for the TCI C++ mapping, in order to avoid conflicts with the different names used, for example, in the C mapping.

C++ class identifiers are omitting the trailing "Type" at the end of the abstract definitions, e.g. the type `TciModuleIdType` is mapped to `TciModuleId` in C++.

10.3 Memory management

A general policy for memory management is not defined in this mapping. However, parameters are passed as pointers (or references) where possible, and a clone method has been added to the definition of every interface. The clone method can be used by the receiving entity to make a local copy where needed.

10.4 Error handling

No additional error handling has been defined for this mapping.

10.5 Type mapping

This clause introduces the TRI C++ language mapping for the abstract types defined in 7.2. The following concepts have been used:

- Pure virtual classes have been used, following the concept of an interface.
- The Standard Template Library (STL) has been used as it is a standardized way of using container classes, and iterators, such as lists. All classes define the operator "<" for easy insertion in STL containers.
- C++ types have been encapsulated under abstract definitions, like `Tfloat` or `Tinteger`.

10.5.1 Encapsulated C++ types

The following types have been defined in order to keep the definitions of data types and operations as general as possible:

```

Boolean type definition:      typedef bool Tboolean
Integer type definition:     typedef long int Tinteger
Size type definition:        typedef unsigned long int Tsize
Index type definition        typedef unsigned long int Tindex
Float type definition:       typedef double Tfloat
Char type definition:        typedef char Tchar
String type definition:      typedef std::string Tstring
Universal char type definition: typedef wchar_t TuniversalChar
Universal string type definition: typedef std::wstring TuniversalString
Bit type definition:         typedef unsigned char Tbit

```

10.5.2 General abstract data types

10.5.2.1 TciBehaviourId

Identifies a TTCN-3 behaviour functions. It is mapped to the following pure virtual class:

```

class TciBehaviourId: ORG\_ETSI\_TTCN3\_TRI::QualifiedName {
public:
    virtual ~TciBehaviourId ();
    virtual Tboolean equals (const TciBehaviourId &bid) const =0;
    virtual TciBehaviourId \* cloneBehaviourId () const =0;
    virtual Tboolean operator< (const TciBehaviourId &bid) const =0;
}

```


10.5.2.1.1 Methods

- [~TciBehaviourId](#) ()
Destructor
- [equals](#) (const [TciBehaviourId](#) &bid)
Returns true if both objects are equal.
- [cloneBehaviourId](#) ()
Return a copy of the [TciBehaviourId](#).
- [operator<](#) (const [TciBehaviourId](#) &bid)
Operator < overload.

10.5.2.2 TciModuleId

A value of TciModuleId specifies the name of a TTCN-3 module as defined in the ATS. It is mapped to the following pure virtual class:

```
class TciModuleId {
public:
    virtual ~TciModuleId ();
    virtual const Tstring & getObjectName() const = 0;
    virtual void setObjectName (const Tstring &p_name)=0;
    virtual Tboolean equals (const TciModuleId &mid) const =0;
    virtual TciModuleId * cloneModuleId () const =0;
    virtual Tboolean operator< (const TciModuleId &mid) const =0;
}
```

10.5.2.2.1 Methods

- [~TciModuleId](#) ()
Destructor
- getObjectName ()
Get the moduleId name
- setObjectName (const Tstring &p_name)
Set the moduleId name
- [equals](#) (const [TciModuleId](#) &mid)
Returns true if both objects are equal.
- [cloneModuleId](#) ()
Return a copy of the [TciModuleId](#).
- [operator<](#) (const [TciModuleId](#) &mid)
Operator < overload.

10.5.2.3 TciModuleParameterId

A value of TciModuleParameterId specifies the name of a TTCN-3 module parameter as defined in the ATS. It is mapped to the following pure virtual class:

```
class TriModuleParameterId : ORG_ETSI_TTCN3_TRI::QualifiedNames {
public:
    virtual ~TciModuleParameterId ();
    virtual Tboolean equals (const TciModuleParameterId &mparId) const =0;
    virtual TciModuleParameterId * cloneModuleParameterId () const =0;
    virtual Tboolean operator< (const TciModuleParameterId &mparId) const =0;
}
```

10.5.2.3.1 Methods

- [~TciModuleParameterId](#) ()
Destructor
- [equals](#) (const [TciModuleParameterId](#) &mparId)
Returns true if both objects are equal.

- [cloneModuleParameterId](#) ()
Return a copy of the [TciModuleParameterId](#).
- [operator<](#) (const [TciModuleParameterId](#) &mparId)
Operator < overload.

10.5.2.4 TciTestCaseId

A value of TciModuleParameterId specifies the name of a TTCN-3 testcase as defined in the ATS. It is mapped to the following pure virtual class:

```
class TciTestCaseId : TciBehaviourId {
public:
    virtual ~TciTestCaseId();
    virtual Tboolean equals (const TciTestCaseId &tcid) const =0;
    virtual TciTestCaseId * cloneTestCaseId () const =0;
    virtual Tboolean operator< (const TciTestCaseId &tcid) const =0;
}
```

10.5.2.4.1 Methods

- [~TciTestCaseId](#) ();
Destructor
- [equals](#) (const [TciTestCaseId](#) &tcid)
Returns true if both objects are equal.
- [cloneTestCaseId](#) ()
Return a copy of the [TciTestCaseId](#).
- [operator<](#) (const [TciTestCaseId](#) &tcid)
Operator < overload.

10.5.2.5 TciModuleIdList

A value of TciModuleIdList defines a list of TciModuleId elements. It is mapped to the following pure virtual class:

```
class TciModuleIdList {
public:
    virtual ~TciModuleIdList();
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector< const TciModuleId * > & getComponents () const =0;
    virtual const TciModuleId &get (Tsize p_index) const =0;
    virtual void clear ()=0;
    virtual void add (const TciModuleId &comp)=0;
    virtual Tboolean equals (const TciModuleIdList &midList) const =0;
    virtual TciModuleIdList * cloneModuleIdList () const =0;
    virtual Tboolean operator< (const TciModuleIdList &midList) const =0;
}
```

10.5.2.5.1 Methods

- [~TciModuleIdList](#) ()
Destructor
- [size](#) ()
Return the size of the list
- [isEmpty](#) ()
Return true if the list is empty
- [getComponents](#) ()
Return the elements of the list
- [get](#) (Tindex p_index)
Return the requested element.
- [clear](#) ()
Remove all the components from this list.

- [add](#) (const [TciModuleId](#) &comp)
Add a component to the end of this list
- [equals](#) (const [TciModuleIdList](#) &midList)
Returns true if both objects are equal.
- [cloneModuleIdList](#) ()
Return a copy of the [TciModuleId](#).
- [operator<](#) (const [TciModuleIdList](#) &midList)
Operator < overload.

10.5.2.6 TciModuleParameter

This abstract type is used to represent the parameter name and the default value of a module parameter. It is mapped to the following pure virtual class:

```
class TciModuleParameter {
public:
    virtual ~TciModuleParameter ();
    virtual const TciValue & getDefaultValue () const =0;
    virtual const Tstring & getModuleParameterName () const =0;
    virtual const TciModuleParameterId & getTciModuleParameterId () const =0;
    virtual Tboolean equals (const TciModuleParameter &mpar) const =0;
    virtual TciModuleParameter * cloneModuleParameter () const =0;
    virtual Tboolean operator< (const TciModuleParameter &mpar) const =0;
}
```

10.5.2.6.1 Methods

- [~TciModuleParameter](#) ()
Destructor
- [getDefaultValue](#) ()
Return default value of the parameter
- [getModuleParameterName](#) ()
Return parameter name
- [getTciModuleParameterId](#) ()
Get the name of the module parameter as defined in the ATS.
- [equals](#) (const [TciModuleParameter](#) &mpar)
Returns true if both objects are equal.
- [cloneModuleParameter](#) ()
Return a copy of the [TciModuleParameter](#).
- [operator<](#) (const [TciModuleParameter](#) &mpar)
Operator < overload.

10.5.2.7 TciModuleParameterList

A value of TciModuleParameterList is a list of TciModuleParameter elements. It is mapped to the following pure virtual class:

```
class TciModuleParameterList {
public:
    virtual ~TciModuleParameterList ();
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector<const TciModuleParameter *> & getComponents () const =0;
    virtual const TciModuleParameter *& get (Tindex p_index) const =0;
    virtual void clear ()=0;
    virtual void add (const TciModuleParameter &comp)=0;
    virtual Tboolean equals (const TciModuleParameterList &mparList) const =0;
    virtual TciModuleParameterList * cloneModuleParameterList () const =0;
    virtual Tboolean operator< (const TciModuleParameterList &mparList) const =0;
}
```

10.5.2.7.1 Methods

- `~TciModuleParameterList ()`
Destructor
- `size ()`
Return the size of the list
- `isEmpty ()`
Return true if the list is empty
- `getComponents ()`
Retrieve the vector of module parameters
- `get (Tindex p_index)`
Retrieve the specified element
- `clear ()`
Remove all components from this list.
- `add (const TciModuleParameter &comp)`
Add a component to the end of this list
- `equals (const TciModuleParameterList &mparList)`
Returns true if both objects are equal.
- `cloneModuleParameterList ()`
Return a copy of the `TciModuleParameterList`.
- `operator< (const TciModuleParameterList &mparList)`
Operator < overload.

10.5.2.8 TciParameterPassingMode

Defines the parameter passing mode (IN,OUT,INOUT). It is mapped to the following class:

```
class TciParameterPassingMode {
public:
    static const TciParameterPassingMode IN;
    static const TciParameterPassingMode OUT;
    static const TciParameterPassingMode INOUT;
    ~TciParameterPassingMode();
    TciParameterPassingMode( const TciParameterPassingMode &);
    const Tboolean equals (const TciParameterPassingMode &) const;
    Tstring toString () const;
    Tinteger getParameterPassingMode () const;
}
```

10.5.2.8.1 Methods

- `~TciParameterPassingMode ()`
Destructor
- `TciParameterPassingMode(const TciParameterPassingMode &);`
Copy constructor
- `equals (const TciParameterPassingMode &)`
Returns true if both `TciParameterPassingMode` objects are equal
- `toString ()`
Returns the sting representation of the object
- `getParameterPassingMode ()`
Returns the int value that represents the enumerated value.

10.5.2.8.2 Constants

- `TciParameterPassingMode IN`
Enumerated value IN
- `TciParameterPassingMode OUT`
Enumerated value OUT

- [TciParameterPassingMode](#) `INOUT`
Enumerated value `INOUT`

10.5.2.9 TciParameter

Includes a TTCN-3 Value and a `TciParameterPassingMode`. It is mapped to the following pure virtual class:

```
class TciParameter {
public:
    virtual ~TciParameter ();
    virtual const TciValue & getValue () const =0;
    virtual void setValue (TciValue &value)=0;
    virtual const TciParameterPassingMode &getParameterPassingMode () const =0;
    virtual void setParameterPassingMode (const TciParameterPassingMode &mode)=0;
    virtual const Tstring & getParameterName () const =0;
    virtual void setParameterName (const Tstring &name)=0;
    virtual Tboolean equals (const TciParameter &param) const =0;
    virtual TciParameter * cloneParameter () const =0;
    virtual Tboolean operator< (const TciParameter &param) const =0;
}
```

10.5.2.9.1 Methods

- [~TciParameter](#) ()
Destructor
- [getValue](#) ()
Retrieve the TTCN-3 value
- [setValue](#) ([TciValue](#) &value)
Set the TTCN-3 value
- [TciParameterPassingMode](#) &[getParameterPassingMode](#) ()
Return the parameter passing mode
- [setParameterPassingMode](#) (const [TciParameterPassingMode](#) &mode)
Set the parameter passing mode
- [getParameterName](#) ()
Return the name of the parameter
- [setParameterName](#) (const Tstring &name)
Set the name of the parameter
- [equals](#) (const [TciParameter](#) ¶m)
Returns true if both objects are equal.
- [cloneParameter](#) ()
Return a copy of the [TciParameter](#).
- [operator<](#) (const [TciParameter](#) ¶m)
Operator < overload.

10.5.2.10 TciParameterList

Defines a list of `TciParameter` elements. It is mapped to the following pure virtual class:

```
class TciParameterList {
public:
    virtual ~TciParameterList ();
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector< const TciParameter * > & getComponents () const =0;
    virtual const TciParameter & get (Tindex p_index) const =0;
    virtual void clear ()=0;
    virtual void add (const TciParameter &comp)=0;
    virtual Tboolean equals (const TciParameterList &param) const =0;
    virtual TciParameterList * cloneParameterList () const =0;
    virtual Tboolean operator< (const TciParameterList &param) const =0;
}
```

10.5.2.10.1 Methods

- [~TciParameterList](#) ()
Destructor
- [size](#) ()
Return the size of the list
- [isEmpty](#) ()
Return true if the list is empty
- [getComponents](#) ()
Return a vector with the elements of the list
- [get](#) (Tindex p_index)
Set the specified element.
- [clear](#) ()
Remove all the components from this list.
- [add](#) (const [TciParameter](#) &comp)
Add a component to the end of this list
- [equals](#) (const [TciParameterList](#) ¶m)
Returns true if both objects are equal.
- [cloneParameterList](#) ()
Return a copy of the [TciParameterList](#).
- [operator<](#) (const [TciParameterList](#) ¶m)
Operator < overload.

10.5.2.11 TciParameterType

Includes a TTCN-3 Type and a TciParameterPassingMode. It is mapped to the following pure virtual class:

```
class TciParameterType {
public:
    virtual ~TciParameterType ();
    virtual const TciType & getType () const =0;
    virtual const TciParameterPassingMode & getParameterPassingMode () const =0;
    virtual Tboolean equals (const TciParameterType &parType) const =0;
    virtual TciParameterType * cloneParameterType () const =0;
    virtual Tboolean operator< (const TciParameterType &parType) const =0;
}
```

10.5.2.11.1 Methods

- [~TciParameterType](#) ()
Destructor
- [getType](#) ()
Return the TTCN-3 Type
- [getParameterPassingMode](#) ()
Get the parameter passing mode
- [equals](#) (const [TciParameterType](#) &parType)
Returns true if both objects are equal.
- [cloneParameterType](#) ()
Return a copy of the [TciParameterType](#).
- [operator<](#) (const [TciParameterType](#) &parType)
Operator < overload.

10.5.2.12 TciParameterTypeList

Specifies a list of TciParameterType elements. It is mapped to the following pure virtual class:

```
class TciParameterTypeList {
public:
    virtual ~TciParameterTypeList ();
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector< const TciParameterType * > & getComponents () const =0;
    virtual const TciParameterType & get (Tindex p_position) const =0;
    virtual void clear ()=0;
    virtual void add (const TciParameterType &comp)=0;
    virtual Tboolean equals (const TciParameterTypeList &ptypeList) const =0;
    virtual TciParameterTypeList * cloneParameterTypeList () const =0;
    virtual Tboolean operator< (const TciParameterTypeList &ptypeList) const =0;
}
```

10.5.2.12.1 Methods

- [~TciParameterTypeList](#) ()
Destructor
- [size](#) ()
Return the size of the list
- [isEmpty](#) ()
Returns true if the list is empty
- [getComponents](#) ()
Retrieve a vector which contains the elements of the list
- [get](#) (Tindex p_position)
Return the requested element
- [clear](#) ()
Remove all the components from this list.
- [add](#) (const [TciParameterType](#) &comp)
Add a component to the end of this list
- [equals](#) (const [TciParameterTypeList](#) &ptypeList)
Returns true if both objects are equal.
- [cloneParameterTypeList](#) ()
Return a copy of the [TciParameterTypeList](#).
- [operator<](#) (const [TciParameterTypeList](#) &ptypeList)
Operator < overload.

10.5.2.13 TciTestComponentKind

Represents the kind of different TTCN-3 components. It is mapped to the following class:

```
class TciTestComponentKind {
public:
    static const TciTestComponentKind SYSTEM_COMP;
    static const TciTestComponentKind PTC_COMP;
    static const TciTestComponentKind PTC_ALIVE_COMP;
    static const TciTestComponentKind MTC_COMP;
    static const TciTestComponentKind CTRL_COMP;
    virtual ~TciTestComponentKind ();
    bool equals (TciTestComponentKind &p_other) const;
    Tstring toString () const;
    Tinteger getTestComponentKind() const;
}
```

10.5.2.13.1 Methods

- [~TciTestComponentKind](#) ()
Destructor
- [equals](#) ([TciTestComponentKind](#) &p_other)
Returns true if both objects are equal
- [toString](#) ()
Returns the string representation of the object.
- [getTestComponentKind](#) ()
Return the int value of the enumerated value

10.5.2.13.2 Constants

- [TciTestComponentKind](#) [SYSTEM_COMP](#)
Enumerated value SYSTEM_COMP
- [TciTestComponentKind](#) [PTC_COMP](#)
Enumerated value PTC_COMP
- [TciTestComponentKind](#) [PTC_ALIVE_COMP](#)
Enumerated value PTC_ALIVE_COMP
- [TciTestComponentKind](#) [MTC_COMP](#)
Enumerated value MTC_COMP
- [TciTestComponentKind](#) [CTRL_COMP](#)
Enumerated value CTRL_COMP

10.5.2.14 TciTypeClass

Defines the different type classes in TTCN-3 (i.e. boolean, float, etc.). It is mapped to the following pure virtual class:

```
class TciTypeClass {
public:
    static const TciTypeClass TCI\_ADDRESS
    static const TciTypeClass TCI\_ANYTYPE
    static const TciTypeClass TCI\_BITSTRING
    static const TciTypeClass TCI\_BOOLEAN
    static const TciTypeClass TCI\_CHARSTRING
    static const TciTypeClass TCI\_COMPONENT
    static const TciTypeClass TCI\_ENUMERATED
    static const TciTypeClass TCI\_FLOAT
    static const TciTypeClass TCI\_HEXSTRING
    static const TciTypeClass TCI\_INTEGER
    static const TciTypeClass TCI\_OCTETSTRING
    static const TciTypeClass TCI\_RECORD
    static const TciTypeClass TCI\_RECORD\_OF
    static const TciTypeClass TCI\_ARRAY
    static const TciTypeClass TCI\_SET
    static const TciTypeClass TCI\_SET\_OF
    static const TciTypeClass TCI\_UNION
    static const TciTypeClass TCI\_UNIVERSAL\_CHARSTRING
    static const TciTypeClass TCI\_VERDICT
    ~TciTypeClass ()
    Tboolean equals (const TciTypeClass &p_other) const
    Tstring toString () const
    Tinteger getTypeClass () const
}
```

10.5.2.14.1 Methods

- [~TciTypeClass](#) ()
Destructor
- [equals](#) (const [TciTypeClass](#) &p_other)
Returns true if both objects are equal
- [toString](#) ()
Returns the string representation of the object.

- `getTypeClass()`
Return the int value associated to the enumerated value.

10.5.2.14.2 Constants

- [TciTypeClass TCI_ADDRESS](#)
Type safe enumerated TCI_ADDRESS
- [TciTypeClass TCI_ANYTYPE](#)
Type safe enumerated TCI_ANYTYPE
- [TciTypeClass TCI_BITSTRING](#)
Type safe enumerated TCI_BITSTRING
- [TciTypeClass TCI_BOOLEAN](#)
Type safe enumerated TCI_BOOLEAN
- [TciTypeClass TCI_CHARSTRING](#)
Type safe enumerated TCI_CHARSTRING
- [TciTypeClass TCI_COMPONENT](#)
Type safe enumerated TCI_COMPONENT
- [TciTypeClass TCI_ENUMERATED](#)
Type safe enumerated TCI_ENUMERATED
- [TciTypeClass TCI_FLOAT](#)
Type safe enumerated TCI_FLOAT
- [TciTypeClass TCI_HEXSTRING](#)
Type safe enumerated TCI_HEXSTRING
- [TciTypeClass TCI_INTEGER](#)
Type safe enumerated TCI_INTEGER
- [TciTypeClass TCI_OCTETSTRING](#)
Type safe enumerated TCI_OCTETSTRING
- [TciTypeClass TCI_RECORD](#)
Type safe enumerated TCI_RECORD
- [TciTypeClass TCI_RECORD_OF](#)
Type safe enumerated TCI_RECORD_OF
- [TciTypeClass TCI_ARRAY](#)
Type safe enumerated TCI_ARRAY
- [TciTypeClass TCI_SET](#)
Type safe enumerated TCI_SET
- [TciTypeClass TCI_SET_OF](#)
Type safe enumerated TCI_SET_OF
- [TciTypeClass TCI_UNION](#)
Type safe enumerated TCI_UNION
- [TciTypeClass TCI_UNIVERSAL_CHARSTRING](#)
Type safe enumerated TCI_UNIVERSAL_CHARSTRING
- [TciTypeClass TCI_VERDICT](#)
Type safe enumerated TCI_VERDICT

10.5.3 Abstract TTCN-3 data types and values

10.5.3.1 TciType

A value of TciType represents one of the TTCN-3 types in a TTCN-3 module. It is mapped to the following pure virtual class:

```
class TciType {
public:
    virtual ~TciType ();
    virtual const TciModuleId & getDefiningModule () const =0;
    virtual const Tstring & getName () const =0;
    virtual const TciTypeClass & getTypeClass () const =0;
```

```

virtual const Tstring & getTypeEncoding () const =0;
virtual const Tstring & getTypeEncodingVariant () const =0;
virtual TciValue * newInstance ()=0;
virtual Tboolean equals (const TciType &typ) const =0;
virtual TciType * cloneType () const =0;
virtual Tboolean operator< (const TciType &typ) const =0;
}

```

10.5.3.1.1 Methods

- [~TciType](#) ()
Destructor
- [getDefiningModule](#) ()
Return the defining module as defined in the TTCN-3 ATS.
- [getName](#) ()
Return type name as defined in the ATS.
- [getTypeClass](#) ()
Return this type class
- [getTypeEncoding](#)
Return type encoding as defined in the ATS.
- [getTypeEncodingVariant](#)
Return encoding variant as defined in TTCN-3
- [newInstance](#)
Return type extension
- [equals](#) (const [TciType](#) &typ)
Return true if the types are equals
- [cloneType](#) ()
Return a copy of the [TciType](#).
- [operator<](#) (const [TciType](#) &typ)
Operator < overload.

10.5.3.2 TciValue

A value of TciValue represents TTCN-3 values for a given type. It is mapped to the following pure virtual class:

```

class TciValue {
public:
    virtual ~TciValue ();
    virtual const TciType & getType () const =0;
    virtual const Tstring & getValueEncoding () const =0;
    virtual const Tstring & getValueEncodingVariant () const =0;
    virtual Tboolean notPresent () const =0;
    virtual Tboolean equals (const TciValue &val) const =0;
    virtual TciValue * cloneValue () const =0;
    virtual Tboolean operator< (const TciValue &val) const =0;
}

```

10.5.3.2.1 Methods

- [~TciValue](#) ()
Destructor
- [getType](#) ()
Returns the type of the specified value
- [getValueEncoding](#) ()
Returns the value encoding attribute as defined in TTCN-3
- [getValueEncodingVariant](#) ()
Returns the value encoding variant attribute as defined in TTCN-3
- [notPresent](#) ()
Returns true if the specified value is omit

- [equals](#) (const [TciValue](#) &val)
Returns true if both objects are equal.
- [cloneValue](#) ()
Return a copy of the [TciValue](#).
- [operator<](#) (const [TciValue](#) &val)
Operator < overload.

10.5.3.3 IntegerValue

TTCN-3 integer value support. It is mapped to the following pure virtual class:

```
class IntegerValue : TciValue {
public:
    virtual ~IntegerValue ();
    virtual Tinteger getInt () const =0;
    virtual void setInt (Tinteger p_value)=0;
    virtual Tboolean equals (const IntegerValue &intVal) const =0;
    virtual IntegerValue * cloneIntegerValue () const =0;
    virtual Tboolean operator< (const IntegerValue &intVal) const =0;
}
```

10.5.3.3.1 Methods

- [~IntegerValue](#) ()
Destructor
- [getInt](#) ()
Return integer value
- [setInt](#) (Tinteger p_value)
Set integer value
- [equals](#) (const [IntegerValue](#) &intVal)
Returns true if both objects are equal.
- [cloneIntegerValue](#) ()
Return a copy of the [IntegerValue](#).
- [operator<](#) (const [IntegerValue](#) &intVal) const =0
Operator < overload.

10.5.3.4 FloatValue

TTCN-3 float value support. It is mapped to the following pure virtual class:

```
class FloatValue : TciValue {
public:
    virtual ~FloatValue ();
    virtual Tfloat getFloat () const =0;
    virtual void setFloat (Tfloat p_floatValue)=0;
    virtual Tboolean equals (const FloatValue &floatVal) const =0;
    virtual FloatValue * cloneFloatValue () const =0;
    virtual Tboolean operator< (const FloatValue &floatVal) const =0;
}
```

10.5.3.4.1 Methods

- [~FloatValue](#) ()
Destructor
- [getFloat](#) ()
Return the float value
- [setFloat](#) (Tfloat p_floatValue)
Set float value
- [equals](#) (const [FloatValue](#) &floatVal)
Returns true if both objects are equal.

- [cloneFloatValue](#) ()
Return a copy of the [FloatValue](#).
- [operator<](#) (const [FloatValue](#) &floatVal)
Operator < overload.

10.5.3.5 BooleanValue

TTCN-3 boolean values support. It is mapped to the following pure virtual class:

```
class BooleanValue : TciValue {
public:
    virtual ~BooleanValue ();
    virtual Tboolean getBoolean () const =0;
    virtual void setBoolean (Tboolean p_booleanValue)=0;
    virtual Tboolean equals (const BooleanValue &booleanVal) const =0;
    virtual BooleanValue * cloneBooleanValue () const =0;
    virtual Tboolean operator< (const BooleanValue &booleanVal) const =0;
}
```

10.5.3.5.1 Methods

- [~BooleanValue](#) ()
Destructor
- [getBoolean](#) ()
Return the boolean value
- [setBoolean](#) (Tboolean p_booleanValue)
Set the variable to booleanValue
- [equals](#) (const [BooleanValue](#) &booleanVal)
Returns true if both objects are equal.
- [cloneBooleanValue](#) ()
Return a copy of the [BooleanValue](#).
- [operator<](#) (const [BooleanValue](#) &booleanVal)
Operator < overload.

10.5.3.6 CharstringValue

TTCN-3 charstring value support. . It is mapped to the following pure virtual class:

```
class CharstringValue : TciValue {
public:
    virtual ~CharstringValue ();
    virtual Tchar getChar (Tindex p_position) const =0;
    virtual Tsize getLength () const =0;
    virtual const Tstring & getString () const =0;
    virtual void setChar (Tsize p_position, Tchar p_char)=0;
    virtual void setLength (Tsize p_length)=0;
    virtual void setString (const Tstring &p_charValue)=0;
    virtual Tboolean equals (const CharstringValue &charStr) const =0;
    virtual CharstringValue * cloneCharstringValue () const =0;
    virtual Tboolean operator< (const CharstringValue &charStr) const =0;
}
```

10.5.3.6.1 Methods

- [~CharstringValue](#) ()
Destructor
- [getChar](#) (Tindex p_position)
Return the char at the specified position
- [getLength](#) ()
Return length of the string
- [getString](#) ()
Return the value of the string

- [setChar](#) (Tsize p_position, Tchar p_char)
Set the char at the specified position
- [setLength](#) (Tsize p_length)
Set length of the string
- [setString](#) (const Tstring &p_charValue)
Set the value of the string
- [equals](#) (const [CharstringValue](#) &charStr)
Returns true if both objects are equal.
- [CharstringValue](#) * [cloneCharstringValue](#) ()
Return a copy of the [CharstringValue](#).
- [operator<](#) (const [CharstringValue](#) &charStr)
Operator < overload.

10.5.3.7 UniversalCharstringValue

TTCN-3 universal charstring value support. It is mapped to the following pure virtual class:

```
class UniversalCharstringValue : TciValue {
public:
    virtual ~UniversalCharstringValue ();
    virtual TuniversalChar getChar (Tindex p_position) const =0;
    virtual Tsize getLength () const =0;
    virtual const TuniversalString & getString () const =0;
    virtual void setChar (Tindex p_position, const TuniversalChar p_ucValue)=0;
    virtual void setLength (Tsize p_length)=0;
    virtual void setString (const TuniversalString &p_ucValue)=0;
    virtual Tboolean equals (const UniversalCharstringValue &uniCharstr) const =0;
    virtual UniversalCharstringValue * cloneUniversalCharstringValue () const =0;
    virtual Tboolean operator< (const UniversalCharstringValue &uniCharstr) const =0;
}
```

10.5.3.7.1 Methods

- [~UniversalCharstringValue](#) ()
Destructor
- [getChar](#) (Tindex p_position)
Return the requested element
- [getLength](#) ()
Return the length of the universal charstring
- TuniversalString & [getString](#) ()
Return the textual representation of the string
- [setChar](#) (Tindex p_position, const TuniversalChar p_ucValue)
Set the char at the specified position
- [setLength](#) (Tsize p_length)
Set the length of the string
- [setString](#) (const TuniversalString &p_ucValue)
Set the value of the string
- [equals](#) (const [UniversalCharstringValue](#) &uniCharstr)
Returns true if both objects are equal.
- [cloneUniversalCharstringValue](#) ()
Return a copy of the [UniversalCharstringValue](#).
- [operator<](#) (const [UniversalCharstringValue](#) &uniCharstr)
Operator < overload.

10.5.3.8 BitstringValue

TTCN-3 bitstring value support. It is mapped to the following pure virtual class:

```
class BitstringValue : TciValue {
public:
    virtual ~BitstringValue ()
    virtual Tbit getBit (Tindex p_position) const =0
    virtual Tsize getLength () const =0
    virtual const Tstring & getString () const =0
    virtual void setBit (Tindex p_position, Tbit p_bsValue)=0
    virtual void setLength (Tindex p_new_length)=0
    virtual void setString (const Tstring &p_bsValue)=0
    virtual Tboolean equals (const BitstringValue &bitStr) const =0
    virtual BitstringValue * cloneBitstringValue () const =0
    virtual Tboolean operator< (const BitstringValue &bitStr) const =0
}
```

10.5.3.8.1 Methods

- [~BitstringValue](#) ()
Destructor
- [getBit](#) (Tindex p_position)
Returns the bit at the specified position
- [getLength](#) ()
Returns the length of the string
- [getString](#) ()
Set the value of the string
- [setBit](#) (Tindex p_position, Tbit p_bsValue)
Set the bit value at the specified position
- [setLength](#) (Tindex p_new_length)
Set the length of the string
- [setString](#) (const Tstring &p_bsValue)
Set the string value
- [equals](#) (const [BitstringValue](#) &bitStr)
Returns true if both objects are equal.
- [cloneBitstringValue](#) ()
Return a copy of the [BitstringValue](#).
- [operator<](#) (const [BitstringValue](#) &bitStr)
Operator < overload.

10.5.3.9 OctetstringValue

TTCN-3 octetstring value support. It is mapped to the following pure virtual class:

```
class OctetstringValue : TciValue {
public:
    virtual ~OctetstringValue ();
    virtual Tsize getLength () const =0;
    virtual const Tchar getOctet (Tindex p_position) const =0;
    virtual const Tstring & getString () const =0;
    virtual void setLength (Tsize p_length)=0;
    virtual void setOctet (Tindex p_position, Tchar p_ochar)=0;
    virtual void setString (const Tstring &p_osValue)=0;
    virtual Tboolean equals (const OctetstringValue &octStr) const =0;
    virtual OctetstringValue * cloneOctetstringValue () const =0;
    virtual Tboolean operator< (const OctetstringValue &octStr) const =0;
}
```

10.5.3.9.1 Methods

- [~OctetstringValue](#) ()
Destructor
- [getLength](#) ()
Return the length of the string
- [getOctet](#) (Tindex p_position)
Return the textual representation of the octetchar at the specified position
- [getString](#) ()
Set the string value
- [setLength](#) (Tsize p_length)
Set the length of the string
- [setOctet](#) (Tindex p_position, Tchar p_ochar)
Set the char at specified position
- [setString](#) (const Tstring &p_osValue)
Set the value of the string.
- [equals](#) (const [OctetstringValue](#) &octStr)
Returns true if both objects are equal.
- [cloneOctetstringValue](#) ()
Return a copy of the [OctetstringValue](#).
- [operator<](#) (const [OctetstringValue](#) &octStr)
Operator < overload.

10.5.3.10 HexstringValue

TTCN-3 hexstring value support. It is mapped to the following pure virtual class:

```
class HexstringValue : TciValue {
public:
    virtual ~HexstringValue ();
    virtual Tchar getHex (Tindex p_position) const =0;
    virtual Tsize getLength () const =0;
    virtual const Tstring & getString () const =0;
    virtual void setHex (Tindex p_position, Tchar p_hcValue)=0;
    virtual void setLength (Tsize p_length)=0;
    virtual void setString (const Tstring &p_hsValue)=0;
    virtual Tboolean equals (const HexstringValue &hexStr) const =0;
    virtual HexstringValue * cloneHexstringValue () const =0;
    virtual Tboolean operator< (const HexstringValue &hexStr) const =0;
}
```

10.5.3.10.1 Methods

- [~HexstringValue](#) ()
Destructor
- [getHex](#) (Tindex p_position)
Return the element at the specified position
- [getLength](#) ()
Return the length of the string
- [getString](#) ()
Return the string value
- [setHex](#) (Tindex p_position, Tchar p_hcValue)
Set the hex value at the specified position
- [setLength](#) (Tsize p_length)
Set the length of the string
- [setString](#) (const Tstring &p_hsValue)
Set the value of the string
- [equals](#) (const [HexstringValue](#) &hexStr)
Returns true if both objects are equal.

- [cloneHexStringValue](#) ()
Return a copy of the [HexStringValue](#).
- [operator<](#) (const [HexStringValue](#) &hexStr)
Operator < overload.

10.5.3.11 RecordValue

TTCN-3 record value support. It is mapped to the following pure virtual class:

```
class RecordValue : TciValue {
public:
    virtual ~RecordValue ();
    virtual const TciValue &getField (const Tstring &p_field_name) const =0;
    virtual void setField (const Tstring &p_field_name,const TciValue &p_new_value)=0;
    virtual const std::vector< Tstring *> &getFieldNames () const =0;
    virtual void setFieldOmitted (const Tstring &fieldName)=0;
    virtual Tboolean equals (const RecordValue &rec) const =0;
    virtual RecordValue * cloneRecordValue () const =0;
    virtual Tboolean operator< (const RecordValue &rec) const =0;
}
```

10.5.3.11.1 Methods

- [~RecordValue](#) ()
Destructor
- [getField](#) (const Tstring &p_field_name)
Return a reference to the field name
- [setField](#) (const Tstring &p_field_name,const TciValue &p_new_value)
Set the value of a field
- [getFieldNames](#) ()
Return a list which containing the names of all the fields.
- [setFieldOmitted](#) (const Tstring &fieldName)
Set omit in one field
- [equals](#) (const RecordValue &rec)
Returns true if both objects are equal.
- [cloneRecordValue](#) ()
Return a copy of the [RecordValue](#).
- [operator<](#) (const RecordValue &rec)
Operator < overload.

10.5.3.12 RecordOfValue

TTCN-3 record of value support. It is mapped to the following pure virtual class:

```
class RecordOfValue : TciValue {
public:
    virtual ~RecordOfValue ();
    virtual const TciValue & getField (Tindex p_position)=0;
    virtual void setField (Tindex p_position, const TciValue &p_value)=0;
    virtual void appendField (const TciValue &p_value)=0;
    virtual const TciType & getElementType () const =0;
    virtual Tsize getLength () const =0;
    virtual void setLength (Tsize p_length)=0;
    virtual Tindex getOffset() const =0;
    virtual Tboolean equals (const RecordOfValue &recOf) const =0;
    virtual RecordOfValue * cloneRecordOfValue () const =0;
    virtual Tboolean operator< (const RecordOfValue &recOf) const =0;
}
```


10.5.3.12.1 Methods

- [~RecordOfValue](#) ()
Destructor
- [getField](#) (Tindex p_position)
Return the field at the specified position.
- [setField](#) (Tindex p_position, const [TciValue](#) &p_value)
Set the value at the specified position
- [appendField](#) (const [TciValue](#) &p_value)
Add a value at the end of the record of
- [getElementType](#) ()
Return the type of the elements of this record of
- [getLength](#) ()
Return the length of the object
- [setLength](#) (Tsize p_length)
Set length of the record of
- [getOffset](#) ()
For arrays, return the lower index bound used in the type definition of arrays. Return 0 for record of and set of.
- [equals](#) (const [RecordOfValue](#) &recOf)
Returns true if both objects are equal.
- [cloneRecordOfValue](#) ()
Return a copy of the [RecordOfValue](#).
- [operator<](#) (const [RecordOfValue](#) &recOf)
Operator < overload.

10.5.3.13 UnionValue

TTCN-3 union value support. It is mapped to the following pure virtual class:

```
class UnionValue : TciValue {
public:
    virtual ~UnionValue ()
    virtual void setVariant (const Tstring &p_variantName, const TciValue &p_value)=0;
    virtual const TciValue & getVariant (const Tstring &p_variantName) const =0;
    virtual const Tstring & getPresentVariantName () const =0;
    virtual const std::set< Tstring *> & getVariantNames () const =0;
    virtual Tboolean equals (const UnionValue &unionVal) const =0;
    virtual UnionValue * cloneUnionValue () const =0;
    virtual Tboolean operator< (const UnionValue &unionVal) const =0;
}
```

10.5.3.13.1 Methods

- [~UnionValue](#) ()
Destructor
- [setVariant](#) (const Tstring &p_variantName, const [TciValue](#) &p_value)
Set the variant name to a value
- [getVariant](#) (const Tstring &p_variantName)
Return the value of the variant if exists
- [getPresentVariantName](#) ()
Return the name of the current variant value. null if no initialized.
- [getVariantNames](#) ()
Return a list which contains the variant names as defined in the ATS.
- [equals](#) (const [UnionValue](#) &unionVal)
Returns true if both objects are equal.
- [cloneUnionValue](#) ()
Return a copy of the [UnionValue](#).

- [operator<](#) (const [UnionValue](#) &unionVal)
Operator < overload.

10.5.3.14 EnumeratedValue

TTCN-3 enumerated value support. It is mapped to the following pure virtual class:

```
class EnumeratedValue : TciValue {
public:
    virtual ~EnumeratedValue ();
    virtual const Tstring & getEnum () const =0;
    virtual void setEnum (const Tstring &p_value)=0;
    virtual Tboolean equals (const EnumeratedValue &enumVal) const =0;
    virtual EnumeratedValue * cloneEnumeratedValue () const =0;
    virtual Tboolean operator< (const EnumeratedValue &enumVal) const =0;
}
```

10.5.3.14.1 Methods

- [~EnumeratedValue](#) ()
Destructor
- [getEnum](#) ()
Return current value
- [setEnum](#) (const Tstring &p_value)
Set the enumeration value
- [equals](#) (const [EnumeratedValue](#) &enumVal)
Returns true if both objects are equal.
- [cloneEnumeratedValue](#) ()
Return a copy of the [EnumeratedValue](#).
- [operator<](#) (const [EnumeratedValue](#) &enumVal)
Operator < overload.

10.5.3.15 VerdictValue

TTCN-3 verdict value support. It is mapped to the following pure virtual class:

```
class VerdictValue : TciValue {
public:
    virtual ~VerdictValue ();
    virtual const VerdictValueEnum & getVerdict () const =0;
    virtual void setVerdict (const VerdictValueEnum & p_enum)=0;
    virtual Tboolean equals (const VerdictValue &verdictVal) const =0;
    virtual VerdictValue * cloneVerdictValue () const =0;
    virtual Tboolean operator< (const VerdictValue &verdictVal) const =0;
}
```

10.5.3.15.1 Methods

- [~VerdictValue](#) ()
Destructor
- [VerdictValueEnum](#) & [getVerdict](#) ()
Return the value of the verdict
- [setVerdict](#) (const [VerdictValueEnum](#) & p_enum)
Set the value of the verdict.
- [equals](#) (const [VerdictValue](#) &verdictVal)
Returns true if both objects are equal.
- [cloneVerdictValue](#) ()
Return a copy of the [VerdictValue](#).
- [operator<](#) (const [VerdictValue](#) &verdictVal)
Operator < overload.

10.5.3.16 VerdictValueEnum

Enumeration of verdict values. It is mapped to the following class:

```
class VerdictValueEnum {
public:
    static const VerdictValueEnum PASS;
    static const VerdictValueEnum FAIL;
    static const VerdictValueEnum ERROR;
    static const VerdictValueEnum INCONC;
    ~VerdictValueEnum ();
    VerdictValueEnum ( const VerdictValueEnum &p_other);
    TString toString () const;
    Tinteger getVerdictValueEnum() const;
    Tboolean equals (const VerdictValueEnum &p_other) const;
}
```

10.5.3.16.1 Methods

- [~VerdictValueEnum \(\)](#)
Destructor
- `VerdictValueEnum (const VerdictValueEnum &p_other);`
Copy constructor.
- [toString \(\)](#)
Return string representation
- `getVerdictValueEnum ()`
Return the int value associated to the enumerated value.
- [equals \(const VerdictValueEnum &p_other\)](#)
Return true if objects are equal.

10.5.3.16.2 Constants

- [VerdictValueEnum PASS](#)
Verdict value PASS
- [VerdictValueEnum FAIL](#)
Verdict value FAIL
- [VerdictValueEnum ERROR](#)
Verdict value ERROR
- [VerdictValueEnum INCONC](#)
Verdict value INCONC

10.5.3.17 AddressValue

TTCN-3 address value support. This implementation is platform-dependent. The AddressValue class contains a typename template TAddress. It is mapped to the following pure virtual class:

```
template <typename TAddress>;
class AddressValue {
public:
    virtual ~AddressValue ();
    virtual const TAddress& getAddress () const =0;
    virtual void setAddress ( const TAddress& T)=0;
    virtual Tboolean equals (const AddressValue &addr) const =0;
    virtual AddressValue * cloneAddressValue () const =0;
    virtual Tboolean operator< (const AddressValue &addr) const =0;
}
```

10.5.3.17.1 Methods

- [~AddressValue \(\)](#)
Destructor
- [getAddress \(\)](#)
Return the value of the address

- [setAddress](#) (const TAddress& T)
Set the value of the address
- [equals](#) (const [AddressValue](#) &addr)
Returns true if both objects are equal.
- [cloneAddressValue](#) ()
Return a copy of the [AddressValue](#).
- [operator<](#) (const [AddressValue](#) &addr)
Operator < overload.

10.5.4 Abstract logging types

10.5.4.1 TciValueTemplate

Interface that defines the concrete operations of the TTCN-3 template. It is mapped to the following pure virtual class:

```
class TciValueTemplate {
public:
    virtual ~TciValueTemplate ();
    virtual Tboolean isOmit () const =0;
    virtual Tboolean isAny () const =0;
    virtual Tboolean isAnyOrOmit () const =0;
    virtual const Tstring & getTemplateDef () const =0;
    virtual Tboolean equals (const TciValueTemplate &vtempl) const =0;
    virtual TciValueTemplate * cloneValueTemplate () const =0;
    virtual Tboolean operator< (const TciValueTemplate &vtempl) const =0;
}
```

10.5.4.1.1 Methods

- [~TciValueTemplate](#) ()
Destructor
- [isOmit](#) ()
Return true if value of template is omit
- [isAny](#) ()
Return true if value of template is any
- [isAnyOrOmit](#) ()
Return true value of template if any or omit
- [getTemplateDef](#) ()
Return the template definition as defined in the ATS
- [equals](#) (const [TciValueTemplate](#) &vtempl)
Returns true if both objects are equal.
- [cloneValueTemplate](#) ()
Return a copy of the [TciValueTemplate](#).
- [operator<](#) (const [TciValueTemplate](#) &vtempl)
Operator < overload.

10.5.4.2 TciNonValueTemplate

Support *all* and *any* matching mechanisms over TTCN-3 components and timers. It is mapped to the following pure virtual class:

```
class TciNonValueTemplate {
public:
    virtual ~TciNonValueTemplate ();
    virtual Tboolean isAny () const =0;
    virtual Tboolean isAll () const =0;
    virtual const Tstring & getTemplateDef () const =0;
    virtual Tboolean equals (const TciNonValueTemplate &nvtempl) const =0;
    virtual TciNonValueTemplate * cloneNonValueTemplate () const =0;
    virtual Tboolean operator< (const TciNonValueTemplate &nvtempl) const =0;
}
```

10.5.4.2.1 Methods

- [~TciNonValueTemplate](#) ()
Destructor
- [isAny](#) ()
Return true if value is any
- [isAll](#) ()
Return true if is value all.
- [getTemplateDef](#) ()
Return template definition as defined in the ATS.
- [equals](#) (const [TciNonValueTemplate](#) &nvtempl)
Returns true if both objects are equal.
- [cloneNonValueTemplate](#) ()
Return a copy of the [TciNonValueTemplate](#).
- [operator<](#) (const [TciNonValueTemplate](#) &nvtempl)
Operator < overload.

10.5.4.3 TciValueList

A list of TciValues. It is mapped to the following pure virtual class:

```
class TciValueList {
public:
    virtual ~TciValueList (void);
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector< const TciValue * > & getComponents () const =0;
    virtual const TciValue & get (Tindex index) const =0;
    virtual void clear ()=0;
    virtual void add (const TciValue &comp)=0;
    virtual Tboolean equals (const TciValueList &valList) const =0;
    virtual TciValueList * cloneValueList () const =0;
    virtual Tboolean operator< (const TciValueList &valList) const =0;
}

```

10.5.4.3.1 Methods

- [~TciValueList](#) ()
Destructor
- [size](#) ()
Return the size of the list
- [isEmpty](#) ()
Return true if the list is empty
- [getComponents](#) ()
Return a vector which contains the values.
- [get](#) (Tindex index)
Return the value at the specified position
- [clear](#) ()
Remove all the elements from this list.
- [add](#) (const [TciValue](#) &comp)
Add an element to the end of this list
- [equals](#) (const [TciValueList](#) &valList)
Returns true if both objects are equal.
- [cloneValueList](#) ()
Return a copy of the [TciValueList](#).
- [operator<](#) (const [TciValueList](#) &valList)
Operator < overload.

10.5.4.4 TciValueDifference

Represents the differences during a match operation. It is mapped to the following pure virtual class:

```
class TciValueDifference {
public:
    virtual ~TciValueDifference ();
    virtual const TciValue & getValue () const =0;
    virtual void setValue (TciValue &val)=0;
    virtual const TciValueTemplate & getTciValueTemplate () const =0;
    virtual void setTciValueTemplate (TciValueTemplate &valT)=0;
    virtual const Tstring & getDescription () const =0;
    virtual void setDescription (const Tstring &descr)=0;
    virtual Tboolean equals (const TciValueDifference &vdiff) const =0;
    virtual TciValueDifference * cloneValueDifference () const =0;
    virtual Tboolean operator< (const TciValueDifference &vdiff) const =0;
}
```

10.5.4.4.1 Methods

- [~TciValueDifference](#) ()
Destructor
- [getValue](#) ()
Return the value definition
- [setValue](#) (TciValue &val)
Set the value definition
- [getTciValueTemplate](#) ()
Return the template definition
- [setTciValueTemplate](#) (TciValueTemplate &valT)
Set the template definition
- [getDescription](#) ()
Return a string which describes the difference.
- [setDescription](#) (const Tstring &descr)
Set description
- [equals](#) (const TciValueDifference &vdiff)
Returns true if both objects are equal.
- [cloneValueDifference](#) ()
Return a copy of the [TciValueDifference](#).
- [operator<](#) (const TciValueDifference &vdiff)
Operator < overload.

10.5.4.5 TciValueDifferenceList

Collection of TciValueDifferences. It is mapped to the following pure virtual class:

```
class TciValueDifferenceList {
public:
    virtual ~TciValueDifferenceList ();
    virtual Tsize size () const =0;
    virtual Tboolean isEmpty () const =0;
    virtual const std::vector< TciValueDifference * > & getComponents () const =0;
    virtual const TciValueDifference & get (Tinteger p_position) const =0;
    virtual void clear ()=0;
    virtual void add (const TciValueDifference &comp)=0;
    virtual Tboolean equals (const TciValueDifferenceList &vdList) const =0;
    virtual TciValueDifferenceList * cloneValueDifferenceList () const =0;
    virtual Tboolean operator< (const TciValueDifferenceList &vdList) const =0;
}
```

10.5.4.5.1 Methods

- [~TciValueDifferenceList](#) ()
Destructor
- [size](#) ()
Return the size of the list
- [isEmpty](#) ()
Return true if this list contains no elements.
- [getComponents](#) ()
Return a vector with the elements.
- [get](#) (Tinteger p_position)
Return the requested difference.
- [clear](#) ()
Remove all the components from this list.
- [add](#) (const [TciValueDifference](#) &comp)
Add a component to the end of the list
- [equals](#) (const [TciValueDifferenceList](#) &vdList)
Returns true if both objects are equal.
- [cloneValueDifferenceList](#) ()
Return a copy of the [TciValueDifferenceList](#).
- [operator<](#) (const [TciValueDifferenceList](#) &vdList)
Operator < overload.

10.5.4.6 ComponentStatus

Type-safe enumeration implementation of [ComponentStatus](#). A value of type [ComponentStatus](#) is either inactiveC, runningC, stoppedC or killedC. This value represents the status of a component. It is mapped to the following class:

```
class ComponentStatus {
public:
    static const ComponentStatus INACTIVE\_C;
    static const ComponentStatus RUNNING\_C;
    static const ComponentStatus STOPPED\_C;
    static const ComponentStatus KILLED\_C;
    virtual ~ComponentStatus ();
    ComponentStatus (const ComponentStatus &);
    Tstring toString () const;
    Tboolean equals (const ComponentStatus &cmps) const;
    Tinteger GetComponentStatus () const;
}
```

10.5.4.6.1 Methods

- [~ComponentStatus](#) ()
Destructor.
- [ComponentStatus](#) (const [ComponentStatus](#) &)
Copy constructor.
- [toString](#) ()
Returns the status as string value.
- [equals](#) (const [ComponentStatus](#) &cmps)
Returns true if objects are equal.
- [GetComponentStatus](#) ()
Returns the status value.

10.5.4.6.2 Constants

- [ComponentStatus](#) [INACTIVE_C](#)
Indicates the inactive status of a component
- [ComponentStatus](#) [RUNNING_C](#)
Indicates the running state of a component

- `ComponentStatus STOPPED_C`
Indicates the stopped status of a component.
- `ComponentStatus KILLED_C`
Indicates the killed status of a component.

10.5.4.7 TimerStatus

Type-safe enumeration implementation of [TimerStatus](#). A value of type [TimerStatus](#) is either `runningT`, `inactiveT`, `expiredT`. This value represents the status of a timer. It is mapped to the following class:

```
class TimerStatus {
public:
    static const TimerStatus RUNNING T;
    static const TimerStatus INACTIVE T;
    static const TimerStatus EXPIRED\_T;
    virtual ~TimerStatus ();
    TimerStatus (const TimerStatus &);
    Tstring toString () const;
    Tboolean equals (const TimerStatus &status) const;
    Tinteger getTimerStatus () const;
}
```

10.5.4.7.1 Methods

- [~TimerStatus](#) ()
Destructor.
- [TimerStatus](#) (const [TimerStatus](#) &)
Copy constructor.
- [toString](#) ()
Returns the status as string value.
- [equals](#) (const [TimerStatus](#) &status)
Returns true if objects are equal.
- [getTimerStatus](#) ()
Returns the status value.

10.5.4.7.2 Constants

- [TimerStatus RUNNING T](#)
Indicates the inactive status of a component
- [TimerStatus INACTIVE T](#)
Indicates the running state of a component
- [TimerStatus EXPIRED_T](#)
Indicates the stopped status of a component.

10.5.4.8 TciStatus

Type-safe enumeration implementation of [TciStatus](#). A value of type [TciStatus](#) is either `TCI_OK` or `TCI_ERROR`. This value represents the success or failure of a TCI operation. It is mapped to the following class:

```
class TciStatus {
public:
    static const TciStatus TCI\_OK;
    static const TciStatus TCI\_ERROR;
    virtual ~TciStatus ();
    TciStatus (const TciStatus &);
    Tstring toString () const;
    Tboolean equals (const TciStatus &status) const;
    Tinteger getStatus () const;
}
```


10.5.4.8.1 Methods

- [~TciStatus](#) ()
Destructor.
- [TciStatus](#) (const [TciStatus](#) &)
Copy constructor.
- [toString](#) ()
Returns the status as string value.
- [equals](#) (const [TciStatus](#) &status)
Returns true if objects are equal.
- [getStatus](#) ()
Returns the status value.

10.5.4.8.2 Constants

- [TciStatus](#) [TCI_OK](#)
Indicates the success of an operation.
- [TciStatus](#) [TCI_ERROR](#)
Indicates the failure of an operation.

10.6 Operations mapping

10.6.1 TCI-TM

10.6.1.1 TciTmRequired

Specifies the operations the TM requires from TE. It is mapped to the following interface:

- //Destructor
- virtual [~TciTmRequired](#) ();
-
- //Selects the indicated module for execution
- virtual void [tciRootModule](#) (const [TciModuleId](#) *moduleName)=0;
-
- //The TE provides to the management a list of imported modules of the root module
- virtual const [TciModuleIdList](#) * [getImportedModules](#) () const =0;
-
- //The TE provides to the management a list of module parameters of the identified module
- virtual const [TciModuleParameterList](#) * [tciGetModuleParameters](#) (const [TciModuleId](#) *moduleName)=0;
-
- //The TE provides to the management a list of test cases
- virtual const [TciTestCaseIdList](#) * [tciGetTestCases](#) () const =0;
-
- //The TE provides to the management a list of parameter types of the given test case
- virtual const [TciParameterTypeList](#) * [tciGetTestCaseParameters](#) (const [TciTestCaseId](#) *testCaseId) const =0;
-
- //The TE provides to the management a list of system ports of the given test case
- virtual const TriPortIdList * [tciGetTestCaseTSI](#) (const [TciTestCaseId](#) &testCaseId) const =0;
-
- //Starts a testcase in the currently selected module with the given parameters
- virtual void [tciStartTestCase](#) (const [TciTestCaseId](#) *testCaseId, const [TciParameterList](#) *parameterList)=0;
-
- //Stops the testcase currently being executed
- virtual void [tciStopTestCase](#) ()=0;
-
- //Starts the control part of the selected module
- virtual const TriComponentId * [tciStartControl](#) ()=0;

-
- //Stops execution of the control part
- virtual void [tciStopControl](#) ()=0;

10.6.1.2 TciTmProvided

Specifies the operation the TM has to provide to the TE. It is mapped to the following interface:

- //Destructor
- virtual [~TciTmProvided](#) ()
-
- //Indicates to the TM that a test case with testCaseId has been started
- virtual void [tciTestCaseStarted](#) (const [TciTestCaseId](#) &testCaseId, const [TciParameterList](#) ¶meterList, const Tfloat &timer)=0;
-
- //Called to indicate that the test case has terminated execution
- virtual void [tciTestCaseTerminated](#) (const [VerdictValue](#) &verdict, const [TciParameterList](#) ¶meterList)=0;
-
- //Called to indicate that the control part of the selected module has just terminated execution
- virtual void [tciControlTerminated](#) ()=0;
-
- //The management provides to the TE a Value for the indicated parameterId
- virtual [TciValue](#) * [tciGetModulePar](#) (const [TciModuleParameterId](#) ¶meterId)=0;
-
- //Indicates the occurrence of an unrecoverable error situation
- virtual void [tciError](#) (const Tstring &message)=0;
-
- //The TE indicates a message of a test component
- virtual void [tciLog](#) (const TriComponentId &testComponentId, const Tstring &message)=0;

10.6.2 TCI-CD

10.6.2.1 TciCdRequired

This class defines the TCI_CD required interface. It is mapped to the following interface:

- //Destructor
- virtual [~TciCdRequired](#) ();
-
- //Returns a type representing a ttcn type
- virtual const [TciType](#) * [getTypeForName](#) (const Tstring typeName) const =0;
-
- //Constructs and returns a basic TTCN-3 integer type
- virtual const [TciType](#) & [getInteger](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 float type
- virtual const [TciType](#) & [getFloat](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 boolean type
- virtual const [TciType](#) & [getBoolean](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 charstring type
- virtual const [TciType](#) & [getCharstring](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 universal charstring type
- virtual const [TciType](#) & [getUniversalCharstring](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 hexstring type
- virtual const [TciType](#) & [getHexstring](#) () const =0;
-
- //Constructs and returns a basic TTCN-3 bitstring type
- virtual const [TciType](#) & [getBitstring](#) () const =0;
-

- //Constructs and returns a basic TTCN-3 octetstring type
- virtual const [TciType](#) & [getOctetstring](#) () const =0
-
- //Constructs and returns a basic TTCN-3 verdict type
- virtual const [TciType](#) & [getVerdict](#) () const =0
-
- //The TE will be notified about an unrecoverable error situation within the CD
- virtual void [tciErrorReg](#) (const Tstring message)=0

10.6.2.2 TciCdProvided

This class defines the TCI_CD provided interface. It is mapped to the following interface:

- //Destructor
- virtual [~TciCdProvided](#) ()
-
- //This operation is called whenever the TE has to decode and encode value
- virtual [TciValue](#) * [decode](#) (const TriMessage *p_message, const [TciType](#) *p_decodingHypothesis)=0
-
- //This operation is called whenever the TE has to encode a Value
- virtual TriMessage * [encode](#) (const [TciValue](#) *p_value)=0

10.6.3 TCI-CH

10.6.3.1 TciChRequired

This class defines the TCI_CH required interface. It is mapped to the following interface:

- //Default destructor
- virtual [~TciChRequired](#) ()
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciSendConnected has been called
- virtual void [tciEnqueueMsgConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const [TciValue](#) *rcvdMessage)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciCallConnected has been called
- virtual void [tciEnqueueCallConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciParameterList](#) *parameterList)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciReplyConnected has been called
- virtual void [tciEnqueueReplyConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciParameterList](#) *parameterList, const [TciValue](#) *returnValue)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciRaiseConnected has been called
- virtual void [tciEnqueueRaiseConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciValue](#) *exception)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciCreateTestComponentReq has been called
- virtual const TriComponentId * [tciCreateTestComponent](#) (const [TciTestComponentKind](#) *kind, const [TciType](#) *componentType, const Tstring *name)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciStartTestComponentReq has been called
- virtual void [tciStartTestComponent](#) (const TriComponentId *component, const [TciBehaviourId](#) *behaviour, const [TciParameterList](#) *parameterList)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciStopTestComponentReq has been called
- virtual void [tciStopTestComponent](#) (const TriComponentId *component)=0
-

- //This operation is called by the CH at the local TE when at a remote TE a provided tciConnect //has been called
- virtual void [tciConnect](#) (const TriPortId *fromPort, const TriPortId *toPort)
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciDisconnect has been called
- virtual void [tciDisconnect](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided tciMapReq //has been called
- virtual void [tciMap](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided tciUnmapReq //has been called
- virtual void [tciUnmap](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided tciUnmapReq //has been called
- virtual void [tciTestComponentTerminated](#) (const TriComponentId *component, const [VerdictValue](#) *verdict) const =0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciTestComponentRunningReq has been called
- virtual Tboolean [tciTestComponentRunning](#) (const TriComponentId *component) const =0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciTestComponentDoneReq has been called
- virtual Tboolean [tciTestComponentDone](#) (const TriComponentId *comp) const =0
-
- //This operation can be called by the CH at the appropriate local TE when at a remote TE a //provided tciGetMTCReq has been called
- virtual const TriComponentId * [tciGetMTC](#) () const =0
-
- //This operation is called by the CH at the appropriate local TE when at a remote TE a provided //tciExecuteTestCaseReq has been called
- virtual void [tciExecuteTestCase](#) (const [TciTestCaseId](#) *testCaseId, const TriPortIdList *tsiPortList)=0
-
- //This operation is called by the CH at appropriate local TEs when at a remote TE a provided //tciResetReq has been called
- virtual void [tciReset](#) ()=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciKillTestComponentReq has been called
- virtual void [tciKillTestComponent](#) (const TriComponentId *comp)=0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciTestComponentAliveReq has been called
- virtual Tboolean [tciTestComponentAlive](#) (const TriComponentId *comp) const =0
-
- //This operation is called by the CH at the local TE when at a remote TE a provided //tciTestComponentKilledReq has been called
- virtual Tboolean [tciTestComponentKilled](#) (const TriComponentId *comp) const =0

10.6.3.2 TciChProvided

This class defines the TCI_CH provided interface. . It is mapped to the following interface:

- //Destructor
- virtual [~TciChProvided](#) ()
-
- //Called by the TE when it executes a TTCN-3 unicast send operation on a component port
- virtual void [tciSendConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const [TciValue](#) *sendMessage)=0
-
- //Called by the TE when it executes a TTCN-3 broadcast send operation on a component port
- virtual void [tciSendConnectedBC](#) (const TriPortId *sender, const [TciValue](#) *sendMessage)=0
-
- //Called by the TE when it executes a TTCN-3 multicast send operation on a component port

- virtual void [tciSendConnectedMC](#) (const TriPortId *sender, const TriComponentIdList *receivers, const [TciValue](#) *sendMessage)=0
-
- //Called by the TE when it executes a TTCN-3 unicast call operation on a component port
- virtual void [tciCallConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciParameterList](#) *parameterList)=0
-
- //Called by the TE when it executes a TTCN-3 broadcast call operation on a component port
- virtual void [tciCallConnectedBC](#) (const TriPortId *sender, const TriSignatureId *signature, const [TciParameterList](#) *parameterList)=0
-
- //Called by the TE when it executes a TTCN-3 multicast call operation on a component port
- virtual void [tciCallConnectedMC](#) (const TriPortId *sender, const TriComponentIdList *receivers, const TriSignatureId *signature, const [TciParameterList](#) *parameterList)=0
-
- //Called by the TE when it executes a TTCN-3 unicast reply operation on a component port
- virtual void [tciReplyConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciParameterList](#) *parameterList, const [TciValue](#) *returnValue)=0
-
- //Called by the TE when it executes a TTCN-3 broadcast reply operation on a component port
- virtual void [tciReplyConnectedBC](#) (const TriPortId *sender, const TriSignatureId *signature, const [TciParameterList](#) *parameterList, const [TciValue](#) *returnValue)=0
-
- //Called by the TE when it executes a TTCN-3 multicast reply operation on a component
- virtual void [tciReplyConnectedMC](#) (const TriPortId *sender, const TriComponentIdList *receivers, const TriSignatureId *signature, const [TciParameterList](#) *parameterList, const [TciValue](#) *returnValue)=0
-
- //Called by the TE when it executes a TTCN-3 unicast raise operation on a component port
- virtual void [tciRaiseConnected](#) (const TriPortId *sender, const TriComponentId *receiver, const TriSignatureId *signature, const [TciValue](#) *exception)=0
-
- //Called by the TE when it executes a TTCN-3 broadcast raise operation on a component port
- virtual void [tciRaiseConnectedBC](#) (const TriPortId *sender, const TriSignatureId *signature, const [TciValue](#) *exception)=0
-
- //Called by the TE when it executes a TTCN-3 multicast raise operation on a component
- virtual void [tciRaiseConnectedMC](#) (const TriPortId *sender, const TriComponentIdList *receiver, const TriSignatureId *signature, const [TciValue](#) *exception)=0
-
- //Called from the TE when a component has to be created
- virtual const TriComponentId * [tciCreateTestComponentReq](#) (const [TciTestComponentKind](#) *kind, const QualifiedName *componentType, const Tstring &name)=0
-
- //Called by the TE when it executes the TTCN-3 start operation
- virtual void [tciStartTestComponentReq](#) (const TriComponentId *component, const [TciBehaviourId](#) *behaviour, const [TciParameterList](#) *parameterList)=0
-
- //Called by the TE when it executes the TTCN-3 stop operation
- virtual void [tciStopTestComponentReq](#) (const TriComponentId *component)=0
-
- //Called by the TE when it executes a TTCN-3 connect operation
- virtual void [tciConnectReq](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //Called by the TE when it executes a TTCN-3 disconnect operation
- virtual void [tciDisconnectReq](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //Called by the TE when it executes a TTCN-3 map operation
- virtual void [tciMapReq](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //Called by the TE when it executes a TTCN-3 unmap operation
- virtual void [tciUnmapReq](#) (const TriPortId *fromPort, const TriPortId *toPort)=0
-
- //Called by the TE when a test component terminates execution
- virtual void [tciTestComponentTerminatedReq](#) (const TriComponentId *component, const [VerdictValue](#) *verdict)=0
-
- //Called by the TE when it executes a TTCN-3 running operation
- virtual Tboolean [tciTestComponentRunningReq](#) (const TriComponentId *component) const =0

-
- //Called by the TE when it executes a TTCN-3 done operation
- virtual Tboolean [tciTestComponentDoneReq](#) (const TriComponentId *comp) const =0
-
- //Called by the TE when it executes a TTCN-3 mtc operation
- virtual const TriComponentId * [tciGetMTCReq](#) () const =0
-
- //Called by the TE immediately before it starts the test case behaviour on the MTC
- virtual void [tciExecuteTestCaseReq](#) (const [TciTestCaseId](#) *testCaseId, const TriPortIdList *tsiPortList)=0
-
- //Called by the TE at any time to reset the test system
- virtual void [tciResetReq](#) ()=0
-
- //Called by the TE when it executes the TTCN-3 kill operation
- virtual void [tciKillTestComponentReq](#) (const TriComponentId *comp)=0
-
- //Called by the TE when it executes the TTCN-3 alive operation
- virtual Tboolean [tciTestComponentAliveReq](#) (const TriComponentId *comp) const =0
-
- //Called by the TE when it executes the TTCN-3 killed operation
- virtual Tboolean [tciTestComponentKilledReq](#) (const TriComponentId *comp) const =0

10.6.4 TCI-TL

10.6.4.1 TciTlProvided

This class defines the TCI_TL provided Tinterface.

- //Default constructor
- [TciTlProvided](#) ()
-
- // Destructor
- virtual [~TciTlProvided](#) ()
-
- //Called by TE to log the execute test case request
- virtual void [tliTcExecute](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const [TciTestCaseId](#) *tcId, const TriParameterList *triPars, const TriTimerDuration *dur)=0
-
- //Called by TE to log the start of a testcase. This event occurs before the testcase is started
- virtual void [tliTcStart](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciTestCaseId](#) *tcId, const [TciParameterList](#) *tciPars, const TriTimerDuration *dur)=0
-
- //Called by TE to log the stop of a testcase
- virtual void [tliTcStop](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c)=0
-
- //Called by TE to log the start of a testcase
- virtual void [tliTcStarted](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciTestCaseId](#) *tcId, const [TciParameterList](#) *tciPars, const TriTimerDuration *dur)=0
-
- //Called by TE to log the termination of a testcase
- virtual void [tliTcTerminated](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciTestCaseId](#) *tcId, const [TciParameterList](#) *tciPars, const [VerdictValue](#) *verdict)=0
-
- //Called by TE to log the start of the control part
- virtual void [tliCtrlStart](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c)=0
-
- //Called by TE to log the stop of the control part. This event occurs after the control has //stopped. If the control is not represented by TRI component, c is null
- virtual void [tliCtrlStop](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c)=0

-
- //Called by TE to log the termination of the control part
- virtual void [tliCtrlTerminated](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c)=0
-
- //Called by TE to log a unicast send operation
- virtual void [tliMSend_m](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const [TciValue](#) *msgValue, const TriAddress *address, const [TciStatus](#) *encoderFailure, const TriMessage *msg, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast send operation
- virtual void [tliMSend_m BC](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const [TciValue](#) *msgValue, const [TciStatus](#) *encoderFailure, const TriMessage *msg, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast send operation
- virtual void [tliMSend_m MC](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const [TciValue](#) *msgValue, const TriAddressList *addresses, const [TciStatus](#) *encoderFailure, const TriMessage *msg, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a unicast send operation
- virtual void [tliMSend_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const [TciValue](#) *msgValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast send operation
- virtual void [tliMSend_c BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const [TciValue](#) *msgValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast send operation
- virtual void [tliMSend_c MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const [TciValue](#) *msgValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log the enqueueing of a message
- virtual void [tliMDetected_m](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriMessage *msg, const TriAddress *address)=0
-
- //Called by CH to log the enqueueing of a message
- virtual void [tliMDetected_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const [TciValue](#) *msgValue)=0
-
- //Called by TE to log the mismatch of a template
- virtual void [tliMMismatch_m](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TciValue](#) *msgValue, const [TciValueTemplate](#) *msgTpl, const [TciValueDifferenceList](#) *diffs, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log the mismatch of a template
- virtual void [tliMMismatch_c](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TciValue](#) *msgValue, const [TciValueTemplate](#) *msgTpl, const [TciValueDifferenceList](#) *diffs, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0
-
- // Called by TE to log the receiving of a message
- virtual void [tliMRecieve_m](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TciValue](#) *msgValue, const [TciValueTemplate](#) *msgTpl, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log the mismatch of a template
- virtual void [tliMReceive_c](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TciValue](#) *msgValue, const [TciValueTemplate](#) *msgTpl, const TriComponentId *fromComp, const [TciNonValueTemplate](#) *fromTpl)=0

-
- //Called by TE to log a unicast call operation
- virtual void [tliPrCall_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const TriAddress *address, const [TciStatus](#) *encoderFailure, const TriParameterList *triPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast call operation
- virtual void [tliPrCall_m_BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciStatus](#) *encoderFailure, const TriParameterList *triPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast call operation
- virtual void [tliPrCall_m_MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const TriAddressList *addresses, const [TciStatus](#) *encoderFailure, const TriParameterList *triPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a unicast call operation
- virtual void [tliPrCall_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast call operation
- virtual void [tliPrCall_c_BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast call operation
- virtual void [tliPrCall_c_MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log the getcall enqueue operation
- virtual void [tliPrGetCallDetected_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const TriParameterList *triPars, const TriAddress *address)=0
-
- //Called by TE to log the getcall enqueue operation
- virtual void [tliPrGetCallDetected_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const [TciParameterList](#) *tciPars)=0
-
- //Called by TE to log the mismatch of a getcall
- virtual void [tliPrGetCallMismatch_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValueDifferenceList](#) *diffs, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log the mismatch of a getcall
- virtual void [tliPrGetCallMismatch_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValueDifferenceList](#) *diffs, const TriComponentId *from, const [TciValueTemplate](#) *fromTpl)=0
-
- //Called by TE to log getting a call
- virtual void [tliPrGetCall_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0

-
- //Called by TE to log getting a call
- virtual void [tliPrGetCall_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0
-
- //Called by TE to log a unicast reply operation
- virtual void [tliPrReply_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *replValue, const TriAddress *address, const [TciStatus](#) *encoderFailure, const [TriParameterList](#) *triPars, const [TriParameter](#) *repl, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log a broadcast reply operation
- virtual void [tliPrReply_m_BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *replValue, const [TciStatus](#) *encoderFailure, const [TriParameterList](#) *triPars, const [TriParameter](#) *repl, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log a multicast reply operation
- virtual void [tliPrReply_m_MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *replValue, const [TriAddressList](#) *addresses, const [TciStatus](#) *encoderFailure, const [TriParameterList](#) *triPars, const [TriParameter](#) *repl, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log a unicast reply operation
- virtual void [tliPrReply_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciValue](#) *parsValue, const [TciValue](#) *replValue, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log a broadcast reply operation
- virtual void [tliPrReply_c_BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TriPortIdList](#) *to, const TriSignatureId *signature, const [TciValue](#) *parsValue, const [TciValue](#) *replValue, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log og a multicast reply operation
- virtual void [tliPrReply_c_MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const [TriPortIdList](#) *to, const TriSignatureId *signature, const [TciValue](#) *parsValue, const [TciValue](#) *replValue, const [TriStatus](#) *transmissionFailure)=0
-
- //Called by TE to log the getreply enqueue operation
- virtual void [tliPrGetReplyDetected_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const [TriParameterList](#) *triPars, const [TriParameter](#) *repl, const [TriAddress](#) *address)=0
-
- //Called by CH to log the getreply enqueue operation
- virtual void [tliPrGetReplyDetected_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *replValue)=0
-
- //Called by TE to log the mismatch of a getreply operation
- virtual void [tliPrGetReplyMismatch_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValue](#) *replValue, const [TciValueTemplate](#) *replyTpl, const [TciValueDifferenceList](#) *diffs, const [TriAddress](#) *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log the mismatch of a getreply operation
- virtual void [tliPrGetReplyMismatch_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValue](#) *replValue, const [TciValueTemplate](#) *replyTpl, const [TciValueDifferenceList](#) *diffs, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0

-
- //Called by TE to log getting a reply
- virtual void [tliPrGetReply_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValue](#) *replValue, const [TciValueTemplate](#) *replyTpl, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log getting a reply
- virtual void [tliPrGetReply_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValueTemplate](#) *parsTpl, const [TciValue](#) *replValue, const [TciValueTemplate](#) *replyTpl, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0
-
- //Called by TE to log a unicast raise operation
- virtual void [tliPrRaise_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriAddress *address, const TriStatus *encoderFailure, const TriException *exc, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast raise operation
- virtual void [tliPrRaise_m BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriStatus *encoderFailure, const TriException *exc, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast raise operation
- virtual void [tliPrRaise_m MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriAddressList *addresses, const TriStatus *encoderFailure, const TriException *exc, const TriStatus *transmissionFailure)=0
- //Called by TE to log a unicast raise operation
- virtual void [tliPrRaise_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a broadcast raise operation
- virtual void [tliPrRaise_c BC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log a multicast raise operation
- virtual void [tliPrRaise_c MC](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortIdList *to, const TriSignatureId *signature, const [TciParameterList](#) *tciPars, const [TciValue](#) *excValue, const TriStatus *transmissionFailure)=0
-
- //Called by TE to log the catch enqueue operation
- virtual void [tliPrCatchDetected_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const TriException *exc, const TriAddress *address)=0
-
- //Called by TE to log the catch enqueue operation
- virtual void [tliPrCatchDetected_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriPortId *from, const TriSignatureId *signature, const [TciValue](#) *excValue)=0
-
- //Called by TE to log the mismatch of a catch operation
- virtual void [tliPrCatchMismatch_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciValue](#) *excValue, const [TciValueTemplate](#) *excTpl, const [TciValueDifferenceList](#) *diffs, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0

-
- //Called by TE to log the mismatch of a catch operation
- virtual void [tliPrCatchMismatch_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciValue](#) *excValue, const [TciValueTemplate](#) *excTpl, const [TciValueDifferenceList](#) *diffs, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0
-
- //Called by TE to log catching an exception
- virtual void [tliPrCatch_m](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciValue](#) *excValue, const [TciValueTemplate](#) *excTpl, const TriAddress *address, const [TciValueTemplate](#) *addressTpl)=0
-
- //Called by TE to log catching an exception
- virtual void [tliPrCatch_c](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature, const [TciValue](#) *excValue, const [TciValueTemplate](#) *excTpl, const TriComponentId *from, const [TciNonValueTemplate](#) *fromTpl)=0
-
- //Called by TE to log the detection of a catch timeout
- virtual void [tliPrCatchTimeoutDetected](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature)=0
-
- //Called by TE to log catching a timeout
- virtual void [tliPrCatchTimeout](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriPortId *at, const TriSignatureId *signature)=0
-
- //Called by TE to log the create component operation
- virtual void [tliCCreate](#) (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp, const Tstring &name, const Tboolean alive)=0
-
- //Called by TE to log the start component operation
- virtual void [tliCStart](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp, const [TciBehaviourId](#) *beh, const [TciParameterList](#) *tciPars)=0
-
- //Called by TE to log the running component operation
- virtual void [tliCRunning](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp, const [ComponentStatus](#) status)=0
-
- //Called by TE to log the alive component operation
- virtual void [tliCAlive](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp, const [ComponentStatus](#) status)=0
-
- //Called by TE to log the stop component operation
- virtual void [tliCStop](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp)=0
-
- //Called by TE to log the kill component operation
- virtual void [tliCKill](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp)=0
-
- //Called by TE to log the mismatch of a done component operation
- virtual void [tliCDoneMismatch](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const TriComponentId *comp, const [TciNonValueTemplate](#) *compTpl)=0
-
- //Called by TE to log the done component operation
- virtual void [tliCDone](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciNonValueTemplate](#) *compTpl)=0
-
- //Called by TE to log the mismatch of a killed component operation
- virtual void [tliCKilledMismatch](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciNonValueTemplate](#) *compTpl)=0

```

•
• //Called by TE to log the killed component operation
• virtual void tliCKilled (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TciNonValueTemplate *compTpl)=0
•
• //Called by TE to log the termination of a component
• virtual void tliCTerminated (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const VerdictValue *verdict)=0
•
• //Called by TE to log the connect operation
• virtual void tliPConnect (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriPortId *port1, const TriPortId *port2)=0
•
• //Called by TE to log the connect operation
• virtual void tliPDisconnect (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriPortId *port1, const TriPortId *port2)=0
•
• //Called by TE to log the map operation
• virtual void tliPMap (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const TriPortId *port1, const TriPortId *port2)=0
•
• //Called by TE to log the unmap operation
• virtual void tliPUnmap (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriPortId *port1, const TriPortId *port2)=0
•
• //Called by TE to log the port clear operation
• virtual void tliPClear (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriPortId *port)=0
•
• //Called by TE to log the port start operation
• virtual void tliPStart (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriPortId *port)=0
•
• //Called by TE to log the port stop operation
• virtual void tliPStop (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const TriPortId *port)=0
•
• //Called by TE to log the port stop operation
• virtual void tliPHalt (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const TriPortId *port)=0
•
• //Called by TE to log the encode operation
• virtual void tliEncode (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TciValue *val, const TciStatus
  *encoderFailure, const TriMessage *msg, const Tstring &codec)=0
•
• //Called by TE to log the decode operation
• virtual void tliDecode (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriMessage *msg, const TciStatus
  *decoderFailure, const TciValue *val, const Tstring &codec)=0
•
• //Called by TE to log the detection of a timeout
• virtual void tliTTimeoutDetected (const Tstring &am, const timeval ts, const Tstring &src,
  const Tinteger line, const TriComponentId *c, const TriTimerId *timer)=0
•
• //Called by TE to log a timeout mismatch
• virtual void tliTTimeoutMismatch (const Tstring &am, const timeval ts, const Tstring &src,
  const Tinteger line, const TriComponentId *c, const TriTimerId *timer, const
  TciNonValueTemplate *timerTpl)=0
•
• //Called by TE to log a timeout match
• virtual void tliTTimeout (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriTimerId *timer, const TciNonValueTemplate
  *timerTpl)=0
•
• //Called by TE to log the start of a timer
• virtual void tliTStart (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriTimerId *timer, const TriTimerDuration
  *dur)=0

```

```

•
• //Called by TE to log the stop of a timer
• virtual void tliTStop (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const TriTimerId *timer, const TriTimerDuration *dur)=0
•
• //Called by TE to log the reading of a timer
• virtual void tliTRead (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const TriTimerId *timer, const TriTimerDuration *elapsed)=0
•
• //Called by TE to log the running timer operation
• virtual void tliTRunning (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const TriTimerId *timer, const TimerStatus status)=0
•
• //Called by TE to log the entering of a scope
• virtual void tliSEnter (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const QualifiedName &name, const TciParameterList
  *tciPars, const Tstring &kind)=0
•
• //Called by TE to log the leaving of a scope
• virtual void tliSLeave (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const QualifiedName &name, const TciParameterList
  *tciPars, const TciValue *returnValue, const Tstring &kind)=0
•
• //Called by TE to log the modification of the value of a variable
• virtual void tliVVar (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const QualifiedName &name, const TciValue *varValue)=0
•
• //Called by TE to log the value of a module parameter
• virtual void tliModulePar (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const QualifiedName &name, const TciValue
  *parValue)=0
•
• //Called by TE to log the value of a module parameter
• virtual void tliGetVerdict (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const VerdictValue *verdict)=0
•
• //Called by TE to log the setverdict operation
• virtual void tliSetVerdict (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const VerdictValue *verdict, const Tstring
  &reason)=0
•
• //Called by TE to log the TTCN-3 statement log
• virtual void tliLog (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger
  line, const TriComponentId *c, const Tstring *log)=0
•
• //Called by TE to log entering an alt
• virtual void tliAEnter (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c)=0
•
• //Called by TE to log leaving an alt
• virtual void tliALeave (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c)=0
•
• //Called by TE to log the nomatch of an alt
• virtual void tliANomatch (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c)=0
•
• //Called by TE to log repeating an alt
• virtual void tliARepeat (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c)=0
•
• //Called by TE to log entering the default section
• virtual void tliADefaults (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c)=0
•
• //Called by TE to log the activation of a default
• virtual void tliAActivate (const Tstring &am, const timeval ts, const Tstring &src, const
  Tinteger line, const TriComponentId *c, const QualifiedName &name, const TciParameterList
  *tciPars, const TciValue *ref)=0

```

-
- //Called by TE to log the deactivation of a default
- virtual void [tliADeactivate](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciValue](#) *ref)=0
-
- //Called by TE to log entering an alt
- virtual void [tliAwait](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c)=0
-
- //Called by TE to log that the component executed an SUT action
- virtual void [tliAction](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const Tstring &action)=0
-
- //Called by TE to log that the component successfully executed a match operation
- virtual void [tliMatch](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciValue](#) &expr, const [TciValueTemplate](#) &tmpl)=0
-
- //Called by TE to log that the component executed a match operation, and a mismatch occurred
- virtual void [tliMatchMismatch](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const [TciValue](#) &expr, const [TciValueTemplate](#) &tmpl, const [TciValueDifferenceList](#) &diffs)=0
-
- //Can be called by the TE to log additional information during test execution
- virtual void [tliInfo](#) (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId *c, const Tinteger level, const Tstring &info)=0

11 W3C XML mapping

11.1 Introduction

This clause introduces the TCI XML mapping [6], [7] and [8] for the logging interface of TCI. The XML mapping for the logging interface defines how the IDL definitions for TCI-TL described in clause 7 are mapped to XML. The complete schema definitions for this mapping are given in annex B.

11.2 Scopes

The IDL module **tciInterface** is mapped to an XML schema with the name space http://uri.etsi.org/ttcn-3/tci/TLI_v3_3_1.xsd

This schema uses further schemas:

- http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd
for the mapping of simple types to XML.
- http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd
for the mapping of structured types to XML.
- http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd
for the mapping of values to XML.
- http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd
for the mapping of templates to XML.
- http://uri.etsi.org/ttcn-3/tci/Events_v3_3_1.xsd
for the mapping of logging events to XML.

11.3 Type mapping

11.3.1 Mapping of simple types

11.3.1.1 TBoolean

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

11.3.1.2 TString

The IDL **TString** type is mapped to the xsd basic type `string`.

11.3.1.3 TInteger

The IDL **TInteger** type is mapped to the xsd basic type `integer`.

11.3.1.4 TriTimerDurationType

The IDL **TriTimerDurationType** type is mapped to the xsd basic type `float`.

11.3.1.5 TciParameterPassingModeType

The IDL **TciParameterPassingModeType** type is mapped to the xsd basic type `string` with enumeration values `'in'`, `'out'` and `'inout'`.

11.3.1.6 TriStatusType

The IDL **TriStatusType** type is mapped to the xsd basic type `string` with enumeration values `'TRI_Ok'` and `'TRI_Error'`.

11.3.1.7 TciStatusType

The IDL **TriStatusType** type is mapped to the xsd basic type `string` with enumeration values `'TCI_Ok'` and `'TCI_Error'`.

11.3.1.8 ComponentStatusType

The IDL **ComponentStatusType** type is mapped to the xsd basic type `string` with enumeration values `'inactiveC'`, `'runningC'`, `'stoppedC'` and `'killedC'`.

11.3.1.9 TimerStatusType

The IDL **TimerStatusType** type is mapped to the xsd basic type `string` with enumeration values `'runningT'`, `'inactiveT'` and `'expiredT'`.

11.3.1.10 PortStatusType

The IDL **PortStatusType** type is mapped to the xsd basic type `string` with enumeration values `'startedP'`, `'haltedP'` and `'stoppedP'`.

11.3.2 Complex type mapping

11.3.2.1 TriPortIdType

TriPortIdType is mapped to the following complex type:

```
<xsd:complexType name="TriPortIdType">
  <xsd:sequence>
    <xsd:element name="comp" type="Types:TriComponentIdType" />
    <xsd:element name="port" type="Types:Port"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `comp` The TRI component identifier.
- `port` The identification of the port.

Attributes:

- none.

11.3.2.2 TriComponentIdType

TriComponentIdType is mapped to the following complex type:

```
<xsd:complexType name="TriComponentIdType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="null" type="Templates:null"/>
      <xsd:element name="id" type="Types:Id"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `id` The identifier of the TRI component.
- `null` The null identifier. To be used if there is no TRI component identifier.

Attributes:

- none.

11.3.2.3 TriComponentIdListType

TriComponentIdListType is mapped to the following complex type:

```
<xsd:complexType name="TriComponentIdListType">
  <xsd:sequence>
    <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `comp` The identifiers of TRI components in that list.

Attributes:

- none.

11.3.2.4 Port

Port is mapped to the following complex type:

```
<xsd:complexType name="Port">
  <xsd:sequence>
    <xsd:element name="id" type="Types:Id"/>
    <xsd:element name="index" type="xsd:int"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `id` The port identifier.
- `port` The port index.

Attributes:

- none.

11.3.2.5 Id

Id is used as identification for components, ports and timers and is mapped to the following complex type:

```
<xsd:complexType name="Id">
  <xsd:sequence>
    <xsd:element name="name" type="SimpleTypes:TString"/>
    <xsd:element name="id" type="SimpleTypes:TString" minOccurs="0"/>
    <xsd:element name="type" type="SimpleTypes:TString" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `name` The name of the component, port or timer.
- `id` The internal representation of the component, port or timer.
- `type` The type of the component, port or timer.

Attributes:

- none.

11.3.2.6 TriMessageType

TriMessageType is mapped to the following complex type:

```
<xsd:complexType name="TriMessageType">
  <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>
```

Elements:

- `val` The encoded message.

Attributes:

- none.

11.3.2.7 TriParameterType

TriParameterType is mapped to the following complex type:

```
<xsd:complexType name="TriParameterType">
  <xsd:sequence>
    <xsd:element name="val" type="xsd:hexBinary"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="SimpleTypes:TString"/>
  <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

Elements:

- val The encoded parameter.

Attributes:

- name The parameter name.
- mode The parameter passing mode.

11.3.2.8 TriParameterListType

TriParameterListType is mapped to the following complex type:

```
<xsd:complexType name="TriParameterListType">
  <xsd:sequence>
    <xsd:element name="par" type="Types:TriParameterType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Sequence of Elements:

- par The parameters in that list.

Attributes:

- none.

11.3.2.9 TriAddressType

TriAddressType is mapped to the following complex type:

```
<xsd:complexType name="TriAddressType">
  <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>
```

Elements:

- val The address value.

Attributes:

- none.

11.3.2.10 TriAddressListType

TriAddressListType is mapped to the following complex type:

```
<xsd:complexType name="TriAddressListType">
  <xsd:sequence>
    <xsd:element name="addr" type="Types:TriAddressType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `addr` The addresses in that list.

Attributes:

- none.

11.3.2.11 TriExceptionType

TriExceptionType is mapped to the following complex type:

```
<xsd:complexType name="TriExceptionType">
  <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>
```

Elements:

- `val` The exception.

Attributes:

- none.

11.3.2.12 TriSignatureIdType

TriSignatureIdType is mapped to the following complex type:

```
<xsd:complexType name="TriSignatureIdType">
  <xsd:attribute name="val" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

- `val` The signature.

Attributes:

- none.

11.3.2.13 TriTimerIdType

TriTimerIdType is mapped to the following complex type:

```
<xsd:complexType name="TriTimerIdType">
  <xsd:sequence>
    <xsd:element name="id" type="Types:Id"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `id` The identification of the timer.

Attributes:

- none.

11.3.2.14 TriTimerDurationType

TriTimerDurationType is mapped to the following simple type:

```
<xsd:simpleType name="TriTimerDurationType">
  <xsd:restriction base="xsd:float"/>
</xsd:simpleType>
```

11.3.2.15 QualifiedName

QualifiedName is used to fully qualify module parameters, variables, etc and is mapped to the following complex type:

```
<xsd:complexType name="QualifiedName">
  <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
  <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

- `moduleName` The module name of the TTCN-3 module.
- `baseName` The name of the object that is fully qualified.

Attributes:

- none.

11.3.2.16 TciBehaviourIdType

TciBehaviourIdType is mapped to the following complex type:

```
<xsd:complexType name="TciBehaviourIdType">
  <xsd:sequence>
    <xsd:element name="name" type="Types:QualifiedName"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `name` The qualified name of the behaviour.

Attributes:

- none.

11.3.2.17 TciTestCaseIdType

TciTestCaseIdType is mapped to the following complex type:

```
<xsd:complexType name="TciTestCaseIdType">
  <xsd:sequence>
    <xsd:element name="name" type="Types:QualifiedName"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements:

- `name` The qualified name of the test case.

Attributes:

- none.

11.3.2.18 TciParameterType

TciParameterType is mapped to the following complex type:

```
<xsd:complexType name="TciParameterType">
  <xsd:sequence>
    <xsd:element name="val" type="Values:Value"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="SimpleTypes:TString"/>
  <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

Elements:

- `val` The encoded parameter.

Attributes:

- `name` The parameter name.
- `mode` The parameter passing mode.

11.3.2.19 TciParameterListType

TciParameterListType is mapped to the following complex type:

```
<xsd:complexType name="TciParameterListType">
  <xsd:sequence>
    <xsd:element name="par" type="Types:TciParameterType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Sequence of Elements:

- `par` The parameters in that list.

Attributes:

- none.

11.3.3 Abstract value mapping

11.3.3.1 Value

Value is mapped to the following complex type:

```
<xsd:complexType name="Value" mixed="true">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:CharstringValue"/>
    <xsd:element name="universal_charstring" type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
    <xsd:element name="set" type="Values:SetValue"/>
    <xsd:element name="set_of" type="Values:SetOfValue"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
    <xsd:element name="union" type="Values:UnionValue"/>
    <xsd:element name="anytype" type="Values:AnytypeValue"/>
    <xsd:element name="address" type="Values:AddressValue"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:attributeGroup name="ValueAtts">
  <xsd:attribute name="name" type="SimpleTypes:TString" use="optional"/>
  <xsd:attribute name="type" type="SimpleTypes:TString" use="optional"/>
  <xsd:attribute name="module" type="SimpleTypes:TString" use="optional"/>
  <xsd:attribute name="annotation" type="SimpleTypes:TString" use="optional"/>
</xsd:attributeGroup>
```

Choice of Elements:

- `integer` An integer value.
- `float` A float value.
- `boolean` A boolean value.
- `verdicttype` A verdicttype value.
- `bitstring` A bitstring value.
- `hexstring` A hexstring value.
- `octetstring` An octetstring value.
- `charstring` A charstring value.
- `universal_charstring` A universal charstring value.
- `record` A record value.
- `record_of` A record of value.
- `array` An array value.
- `set` A set value.
- `set_of` A set of value.
- `enumerated` An enumerated value.
- `union` A union value.
- `anytype` An anytype value.
- `address` An address value.

Attributes:

- `name` The name of the value, if known.
- `type` The type of the value, if known.
- `module` The module of the value, if known.
- `annotation` A helper attribute to provide additional matching/mismatching information, etc.

11.3.3.2 IntegerValue

IntegerValue is mapped to the following complex type:

```
<xsd:complexType name="IntegerValue">
  <xsd:choice >
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Simple Content:

- `value` The integer value as string.
- `null` If no value is given.

- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.3 FloatValue

FloatValue is mapped to the following complex type:

```
<xsd:complexType name="FloatValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Simple Content:

- value The float value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.4 BooleanValue

BooleanValue is mapped to the following complex type:

```
<xsd:complexType name="BooleanValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Simple Content:

- value The boolean value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.5 Void

11.3.3.6 VerdictValue

VerdictValue is mapped to the following complex type:

```
<xsd:complexType name="VerdictValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Simple Content:

- value The verdict value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.7 BitstringValue

BitstringValue is mapped to the following complex type:

```
<xsd:complexType name="BitstringValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Simple Content:

- value The bitstring value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.8 HexstringValue

HexstringValue is mapped to the following complex type:

```
<xsd:complexType name="HexstringValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Simple Content:

- value The hexstring value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.9 OctetstringValue

OctetstringValue is mapped to the following complex type:

```
<xsd:complexType name="OctetstringValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```



```

    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

```

Simple Content:

- value The octetstring value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.10 CharstringValue

CharstringValue is mapped to the following complex type:

```

<xsd:complexType name="CharstringValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

```

Simple Content:

- value The charstring value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.11 UniversalCharstringValue

UniversalCharstringValue is mapped to the following complex type:

```

<xsd:complexType name="UniversalCharstringValue">
  <xsd:simpleContent>
    <xsd:extension base="SimpleTypes:TString">
      <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

```

Simple Content:

- value The universal charstring value as string.
- null If no value is given.
- omit If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.12 RecordValue

RecordValue is mapped to the following complex type:

```
<xsd:complexType name="RecordValue">
  <xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="integer" type="Values:IntegerValue"/>
      <xsd:element name="float" type="Values:FloatValue"/>
      <xsd:element name="boolean" type="Values:BooleanValue"/>
      <xsd:element name="verdicttype" type="Values:VerdictValue"/>
      <xsd:element name="bitstring" type="Values:BitstringValue"/>
      <xsd:element name="hexstring" type="Values:HexstringValue"/>
      <xsd:element name="octetstring" type="Values:OctetstringValue"/>
      <xsd:element name="charstring" type="Values:CharstringValue"/>
      <xsd:element name="universal_charstring"
        type="Values:UniversalCharstringValue"/>
      <xsd:element name="record" type="Values:RecordValue"/>
      <xsd:element name="record_of" type="Values:RecordOfValue"/>
      <xsd:element name="array" type="Values:ArrayValue"/>
      <xsd:element name="set" type="Values:SetValue"/>
      <xsd:element name="set_of" type="Values:SetOfValue"/>
      <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
      <xsd:element name="union" type="Values:UnionValue"/>
      <xsd:element name="anytype" type="Values:AnytypeValue"/>
      <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Sequence of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- anytype An anytype value.
- address An address value.

- null If no field is given.
- omit If the field is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.13 RecordOfValue

RecordOfValue is mapped to the following complex type:

```
<xsd:complexType name="RecordOfValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="charstring" type="Values:CharstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="array" type="Values:ArrayValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="set" type="Values:SetValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="set_of" type="Values:SetOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of Sequence of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.

- `charstring` A charstring value.
- `universal_charstring` A universal charstring value.
- `record` A record value.
- `record_of` A record of value.
- `array` An array value.
- `set` A set value.
- `set_of` A set of value.
- `enumerated` An enumerated value.
- `union` A union value.
- `anytype` An anytype value.
- `address` An address value.
- `null` If no field is given.
- `omit` If the field is omitted.

Attributes:

- The same attributes as those of `Value`.

11.3.3.14 ArrayValue

ArrayValue is mapped to the following complex type:

```
<xsd:complexType name="ArrayValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="charstring" type="Values:CharstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="array" type="Values:ArrayValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="set" type="Values:SetValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="set_of" type="Values:SetOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:choice>
</xsd:complexType>
```

```

    <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

Choice of Sequence of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- anytype An anytype value.
- address An address value.
- null If no field is given.
- omit If the field is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.15 SetValue

SetValue is mapped to the following complex type:

```

<xsd:complexType name="SetValue">
  <xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="integer" type="Values:IntegerValue"/>
      <xsd:element name="float" type="Values:FloatValue"/>
      <xsd:element name="boolean" type="Values:BooleanValue"/>
      <xsd:element name="verdicttype" type="Values:VerdictValue"/>
      <xsd:element name="bitstring" type="Values:BitstringValue"/>
      <xsd:element name="hexstring" type="Values:HexstringValue"/>
      <xsd:element name="octetstring" type="Values:OctetstringValue"/>
      <xsd:element name="charstring" type="Values:CharstringValue"/>
    </xsd:choice>
  </xsd:choice>
</xsd:complexType>

```

```

<xsd:element name="universal_charstring"
  type="Values:UniversalCharstringValue"/>
<xsd:element name="record" type="Values:RecordValue"/>
<xsd:element name="record_of" type="Values:RecordOfValue"/>
<xsd:element name="array" type="Values:ArrayValue"/>
<xsd:element name="set" type="Values:SetValue"/>
<xsd:element name="set_of" type="Values:SetOfValue"/>
<xsd:element name="enumerated" type="Values:EnumeratedValue"/>
<xsd:element name="union" type="Values:UnionValue"/>
<xsd:element name="anytype" type="Values:AnytypeValue"/>
<xsd:element name="address" type="Values:AddressValue"/>
</xsd:choice>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

Sequence of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- anytype An anytype value.
- address An address value.
- null If no field is given.
- omit If the field is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.16 SetOfValue

SetOfValue is mapped to the following complex type:

```
<xsd:complexType name="SetOfValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="charstring" type="Values:CharstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="array" type="Values:ArrayValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="set" type="Values:SetValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="set_of" type="Values:SetOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Choice of Sequence of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.

- `set` A set value.
- `set_of` A set of value.
- `enumerated` An enumerated value.
- `union` A union value.
- `anytype` An anytype value.
- `address` An address value.
- `null` If no field is given.
- `omit` If the field is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.17 EnumeratedValue

EnumeratedValue is mapped to the following complex type:

```
<xsd:complexType name="EnumeratedValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
</xsd:sequence>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

Sequence of Elements:

- `value` The enumeration value.
- `null` If no value is given.
- `omit` If the value is omitted.

Attributes:

- The same attributes as those of Value.

11.3.3.18 UnionValue

UnionValue is mapped to the following complex type:

```
<xsd:complexType name="UnionValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:CharstringValue"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
    <xsd:element name="set" type="Values:SetValue"/>
    <xsd:element name="set_of" type="Values:SetOfValue"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
    <xsd:element name="union" type="Values:UnionValue"/>
  </xsd:choice>
</xsd:complexType>
```



```

    <xsd:element name="anytype" type="Values:AnytypeValue"/>
    <xsd:element name="address" type="Values:AddressValue"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

Choice of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- anytype An anytype value.
- address An address value.

Attributes:

- The same attributes as those of Value.

11.3.3.19 AnytypeValue

AnytypeValue is mapped to the following complex type:

```

<xsd:complexType name="AnytypeValue">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:OctetstringValue"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
  </xsd:choice>
</xsd:complexType>

```

```

<xsd:element name="set" type="Values:SetValue"/>
<xsd:element name="set_of" type="Values:SetOfValue"/>
<xsd:element name="enumerated" type="Values:EnumeratedValue"/>
<xsd:element name="union" type="Values:UnionValue"/>
<xsd:element name="address" type="Values:AddressValue"/>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

Choice of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- address An address value.

Attributes:

- The same attributes as those of Value.

11.3.3.20 AddressValue

AddressValue is mapped to the following complex type:

```

<xsd:complexType name="AddressValue">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:OctetstringValue"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
  </xsd:choice>
</xsd:complexType>

```

```

<xsd:element name="set" type="Values:SetValue"/>
<xsd:element name="set_of" type="Values:SetOfValue"/>
<xsd:element name="enumerated" type="Values:EnumeratedValue"/>
<xsd:element name="union" type="Values:UnionValue"/>
<xsd:element name="anytype" type="Values:AnytypeValue"/>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

Choice of Elements:

- integer An integer value.
- float A float value.
- boolean A boolean value.
- verdicttype A verdicttype value.
- bitstring A bitstring value.
- hexstring A hexstring value.
- octetstring An octetstring value.
- charstring A charstring value.
- universal_charstring A universal charstring value.
- record A record value.
- record_of A record of value.
- array An array of value.
- set A set value.
- set_of A set of value.
- enumerated An enumerated value.
- union A union value.
- anytype An anytype value.

Attributes:

- The same attributes as those of Value.

11.3.4 Abstract logging types mapping

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

11.3.4.1 TciValueTemplate

TciValueTemplate is mapped to the following complex type:

```

<xsd:complexType name="TciValueTemplate">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Values:Value">
      <xsd:choice minOccurs="0">
        <xsd:element name="integer" type="Templates:IntegerTemplate"/>
        <xsd:element name="float" type="Templates:FloatTemplate"/>
        <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
        <xsd:element name="verdicttype" type="Templates:VerdictTemplate"/>
        <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
<xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
<xsd:element name="charstring" type="Templates:CharstringTemplate"/>
<xsd:element name="universal_charstring"
  type="Templates:UniversalCharstringTemplate"/>
<xsd:element name="record" type="Templates:RecordTemplate"/>
<xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
<xsd:element name="array" type="Templates:ArrayTemplate"/>
<xsd:element name="set" type="Templates:SetTemplate"/>
<xsd:element name="set_of" type="Templates:SetOfTemplate"/>
<xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
<xsd:element name="union" type="Templates:UnionTemplate"/>
<xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
<xsd:element name="address" type="Templates:AddressTemplate"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
</xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

```

Choice of Elements:

- integer An integer template.
- float A float template.
- boolean A boolean template.
- verdicttype A verdicttype template.
- bitstring A bitstring template.
- hexstring A hexstring template.
- octetstring An octetstring template.
- charstring A charstring template.
- universal_charstring A universal charstring template.
- record A record template.
- record_of A record of template.
- array An array template.
- set A set template.
- set_of A set of template.
- enumerated An enumerated template.
- union A union template.
- anytype An anytype template.
- address An address template.
- omit An omit template.
- any An any template.
- anyoromit An anyoromit template.
- templateDef A complex template definition.

Attributes:

- none.

11.3.4.2 TciNonValueTemplate

TciNonValueTemplate is mapped to the following complex type:

```
<xsd:complexType name="TciNonValueTemplate">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="any" type="Templates:any"/>
      <xsd:element name="all" type="Templates:all"/>
      <xsd:element name="templateDef" type="SimpleTypes:TString"/>
      <xsd:element name="null" type="Templates:null"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Choice of Elements:

- any An any template.
- all An all template.
- templateDef A complex template definition.
- null No template is given.

Attributes:

- none.

11.3.4.3 TciValueList

TciValueList is mapped to the following complex type:

```
<xsd:complexType name="TciValueList">
  <xsd:sequence>
    <xsd:element name="val" type="Values:Value"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Sequence of Elements:

- val The values in the value list.

Attributes:

- none.

11.3.4.4 TciValueDifference

TciValueDifference is mapped to the following complex type:

```
<xsd:complexType name="TciValueDifference">
  <xsd:sequence>
    <xsd:element name="val" type="SimpleTypes:xpath"/>
    <xsd:element name="tpl"
type="SimpleTypes:xpath"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
  <xsd:attribute name="desc" type="SimpleTypes:TString"
use="optional"/>
</xsd:complexType>
```

Sequence of Elements:

- `val` A reference to the mismatching value.
- `tmpl` A reference to the template.

Attributes:

- The same attributes as those of `Value`.
- `desc` The reason of the mismatch.

11.3.4.5 TciValueDifferenceList

TciValueDifferenceList is mapped to the following complex type:

```
<xsd:complexType name="TciValueDifferenceList">
  <xsd:sequence>
    <xsd:element name="diff" type="Templates:TciValueDifference"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Sequence of Elements:

- `diff` The value/template differences in the value difference list.

Attributes:

- `none`.

11.4 Mapping of the operations on the logging interface

Every operation provided at the logging interface has a corresponding complex type definition in XML. These complex type definitions are extensions of `Event`.

11.4.1 Event

Event is mapped to the following complex type:

```
<!-- common definition for all events -->
<xsd:complexType name="Event" mixed="true">
  <xsd:sequence>
    <xsd:element name="am" type="SimpleTypes:TString"/>
  </xsd:sequence>
  <xsd:attribute name="ts" type="xsd:long" use="required"/>
  <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
  <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

  <!-- general identifier structure for test components, ports and timer -->
  <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
  <xsd:attribute name="id" type="SimpleTypes:TString" use="required"/>
  <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Elements:

- `am` A message, to be used for further information in the log.

Attributes:

- `ts` The time when the event is produced.
- `src` The source file of the test specification.
- `line` The line number where the request is performed.

- name The name of the component which produces this event.
- id The id of the component which produces this event.
- type The type of the component which produces this event.

11.4.2 The TCI-TL interface

11.4.2.1 TCI-TL provided

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

```

<!-- testcases -->
<xsd:complexType name="tliTcExecute">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcStart">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcStop">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcStarted">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcTerminated">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="verdict" type="Values:VerdictValue"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- control -->
<xsd:complexType name="tliCtrlStart">
  <xsd:complexContent>
    <xsd:extension base="Events:Event"/>
  </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:complexType name="tliCtrlStop">
  <xsd:complexContent>
    <xsd:extension base="Events:Event"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCtrlTerminated">
  <xsd:complexContent>
    <xsd:extension base="Events:Event"/>
  </xsd:complexContent>
</xsd:complexType>

<!-- asynchronous communication -->
<xsd:complexType name="tliMSend_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
            <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMSend_m_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMSend_m_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
            <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```



```

    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c_BC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c_MC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMDetected_m">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Types:TriMessageType"/>
          <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMDetected_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMMismatch_m">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="msgTpl" type="Templates:TciValueTemplate"/>
          <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>

```

```

        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMReceive_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMReceive_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- synchronous communication -->
<xsd:complexType name="tliPrCall_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_m_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_m_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
            <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_c_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrCall_c_MC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetCallDetected_m">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="triPars" type="Types:TriParameterListType" minOccurs="0"/>
          <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetCallDetected_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetCallMismatch_m">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
          <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
          <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
          <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetCallMismatch_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
          <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
          <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
          <xsd:element name="fromTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

```

```

</xsd:complexType>

<xsd:complexType name="tliPrGetCall_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCall_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
            <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_m_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

        type="SimpleTypes:TriStatusType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:choice>
  </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_m_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
            minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
              minOccurs="0"/>
            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
            <xsd:element name="addresses" type="Types:TriAddressListType"
              minOccurs="0"/>
            <xsd:element name="transmission-failure"
              type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_c_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrReply_c_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

minOccurs="0"/>
  </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyDetected_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="triPars" type="Types:TriParameterListType" minOccurs="0"/>
        <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyDetected_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>

```

```

        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="replTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m_BC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```



```

<xsd:complexType name="tliPrRaise_m_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
            <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">

```

```

    <xsd:sequence>
      <xsd:element name="at" type="Types:TriPortIdType"/>
      <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
      <xsd:element name="signature" type="Types:TriSignatureIdType"/>
      <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
      <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="excTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="excTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="excValue" type="Values:Value"/>
        <xsd:element name="excTpl" type="Templates:TciValueTemplate"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>

```

```

        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="excTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeoutDetected">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeout">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- components -->
<xsd:complexType name="tliCCreate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="name" type="SimpleTypes:TString"/>
                <xsd:element name="alive" type="SimpleTypes:TBoolean"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStart">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="name" type="Types:TciBehaviourIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCRunning">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCAlive">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStop">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKill">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDoneMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKilledMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDone">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKilled">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCTerminated">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="verdict" type="Values:VerdictValue" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- ports -->
<xsd:complexType name="tliPConnect">

```

```

        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPDisconnect">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPMap">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPUnmap">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortConfiguration"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPClear">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortStatus"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPStart">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortStatus"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPStop">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortStatus"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPHalt">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:PortStatus"/>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- codec -->
    <xsd:complexType name="tliEncode">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="val" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="msg" type="Types:TriMessageType"/>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    </xsd:choice>
                    <xsd:element name="codec" type="SimpleTypes:TString"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliDecode" mixed="true">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="msg" type="Types:TriMessageType"/>
                    <xsd:choice>
                        <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="val" type="Values:Value"/>
                    </xsd:choice>
                    <xsd:element name="codec" type="SimpleTypes:TString"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

```

```

        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- timers -->
    <xsd:complexType name="tliTimeoutDetected">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTimeoutMismatch">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType" />
            <xsd:element name="timerTpl" type="Templates:TciNonValueTemplate" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTimeout">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType" />
            <xsd:element name="timerTpl" type="Templates:TciNonValueTemplate" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTStart">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType"/>
            <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTStop">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType"/>
            <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTRead">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType"/>
            <xsd:element name="elapsed" type="SimpleTypes:TriTimerDurationType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTRunning">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="timer" type="Types:TriTimerIdType"/>
          </xsd:sequence>
          <xsd:attribute name="status" type="SimpleTypes:TimerStatusType"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

```

```

    </xsd:complexContent>
</xsd:complexType>

<!-- scope -->
<xsd:complexType name="tliSEnter">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="kind" type="SimpleTypes:TString"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliSLeave">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="returnValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="kind" type="SimpleTypes:TString"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- variables and module parameter -->
<xsd:complexType name="tliVar">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="val" type="Values:Value" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliModulePar">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="val" type="Values:Value" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- verdicts -->
<xsd:complexType name="tliGetVerdict">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="verdict" type="Values:VerdictValue"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliSetVerdict">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="verdict" type="Values:VerdictValue"/>
        <xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- log -->
<xsd:complexType name="tliLog">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">

```

```

        <xsd:sequence>
            <xsd:element name="log" type="SimpleTypes:TString"/>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- alt -->
<xsd:complexType name="tliAEnter">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliALeave">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliADefaults">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliAActivate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" />
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="ref" type="Values:Value"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliADeactivate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="ref" type="Values:Value"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliANomatch">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliARepeat">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliAwait">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliAction">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="action" type="SimpleTypes:TString"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMatch">
    <xsd:complexContent mixed="true">

```



```

    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="expr" type="Values:Value"/>
        <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMatchMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="expr" type="Values:Value"/>
        <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliInfo">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="level" type="SimpleTypes:TInteger"/>
        <xsd:element name="info" type="SimpleTypes:TString"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

12 Use scenarios

This clause contains use scenarios that should help users of the TCI and tool vendors providing the TCI understand the semantics of the operations defined within the present document.

The scenarios are defined in terms of UML sequence diagrams. The sequence diagram shows the interactions between the TCI entities. The scenarios are explained and where applicable underpinned with a TTCN-3 fragment corresponding to the scenario.

12.1 Initialization, collecting information, logging

12.1.1 Use scenario: initialization

The scenario in figure 9 shows the initialization phase for a test system when a TTCN-3 module is to be selected for execution. At first, a root module has to be set with `tciRootModule`. The module parameters of the root module can be obtained with `tciGetModuleParameters`. Module parameter information can be used to ask the test system user for concrete values for each module parameter. The list of test cases available in the root module can be retrieved with `tciGetTestCases`. These test cases can be directly executed from the test management. Their parameters and their test system interface can be obtained with `tciGetTestCaseParameters` and `tciGetTestCaseTSI`, respectively.

12.1.1.1 Sequence diagram

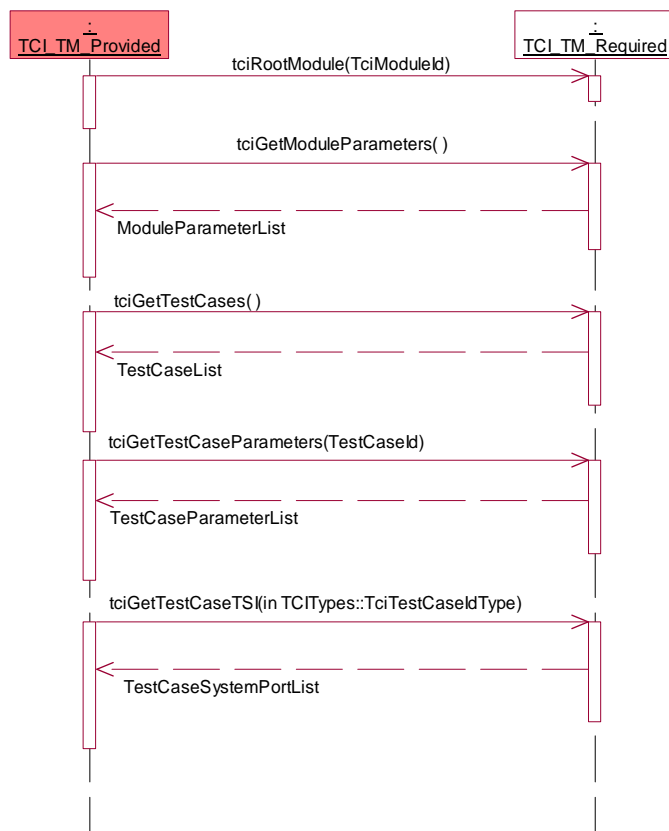


Figure 9: Use scenario - initialization

12.1.1.2 TTCN-3 fragment

The initialization is outside the scope of TTCN-3.

12.1.2 Use scenario: requesting module parameters

The scenario in figure 10 shows how a test component requests the actual value of a module parameter needed for the execution of its test behaviour. At first, the type of a module parameter is requested, then the value can be constructed by the TM and given to the TE.

12.1.2.1 Sequence diagram

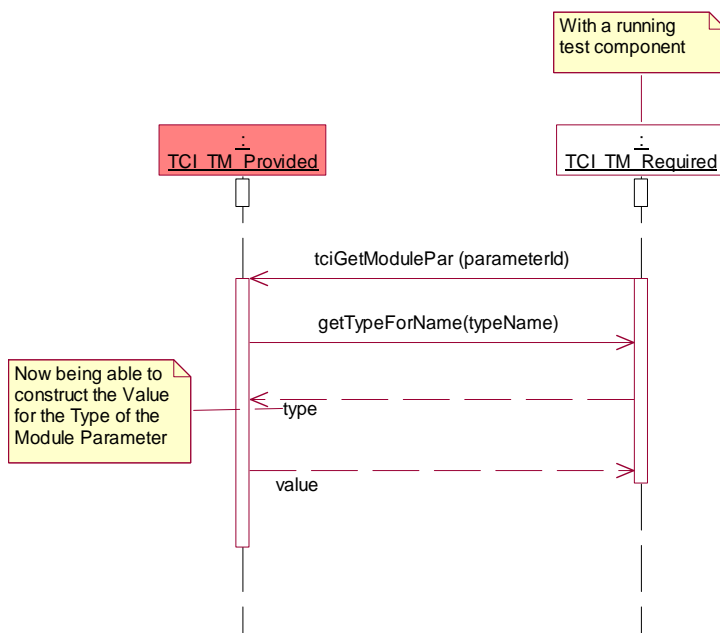


Figure 10: Use scenario - requesting module Pars

12.1.2.2 TTCN-3 fragment

```

module AModule {
  ...
  modulepar {
    integer AModulePar
  }
  ...
  function AFunction (...) ... {
    integer x;
    ...
    x:= 2+AModulePar; // an expression with a module parameter
    ...
  }
  ...
}
  
```

12.1.3 Use scenario: logging

The scenario in figure 11 shows logging of information during the execution of a test behaviour by a test component. The message to be logged is propagated to the test logging.

12.1.3.1 Sequence diagram

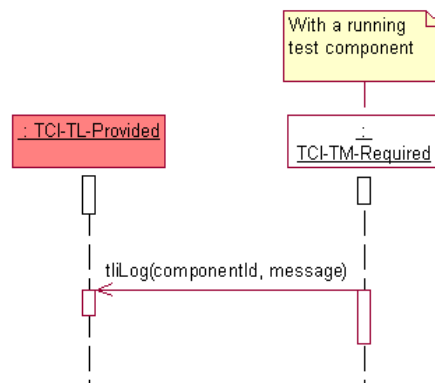


Figure 11: Use scenario - logging

12.1.3.2 TTCN-3 fragment

```

module AModule {
  ...
  function AFunction (...) ... {
    ...
    log('AMessage');
    ...
  }
  ...
}

```

12.2 Execution of test cases and control

12.2.1 Use scenario: execution of control

The scenario in figure 12 shows the sequence of operations to execute the control part of a TTCN-3 module. The module containing the control part is selected first, then the control is started, then it is executed until the execution is terminated by TE.

12.2.1.1 Sequence diagram

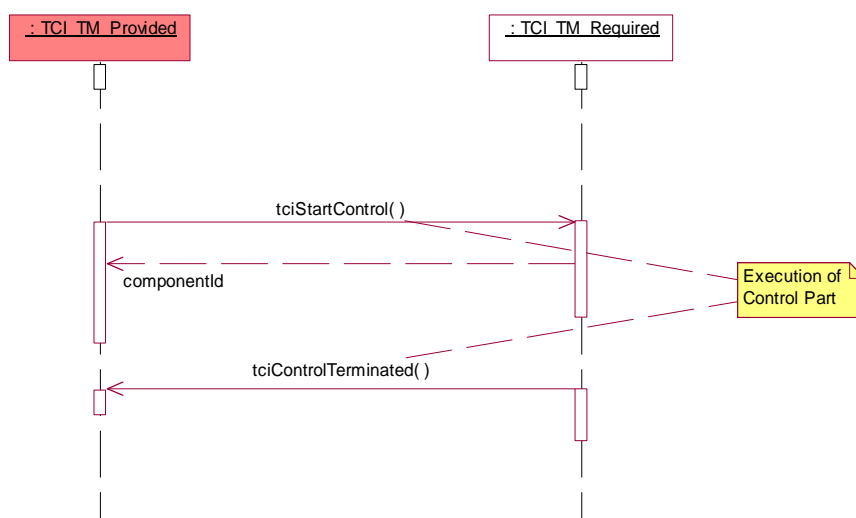


Figure 12: Use scenario - execution of control

12.2.1.2 TTCN-3 fragment

```

module AModule {
  ...
  control {
    ...
  }
  ...
}

```

12.2.2 Use scenario: test case execution within control

The scenario in figure 13 shows how a test case is executed within the control part.

12.2.2.1 Sequence diagram

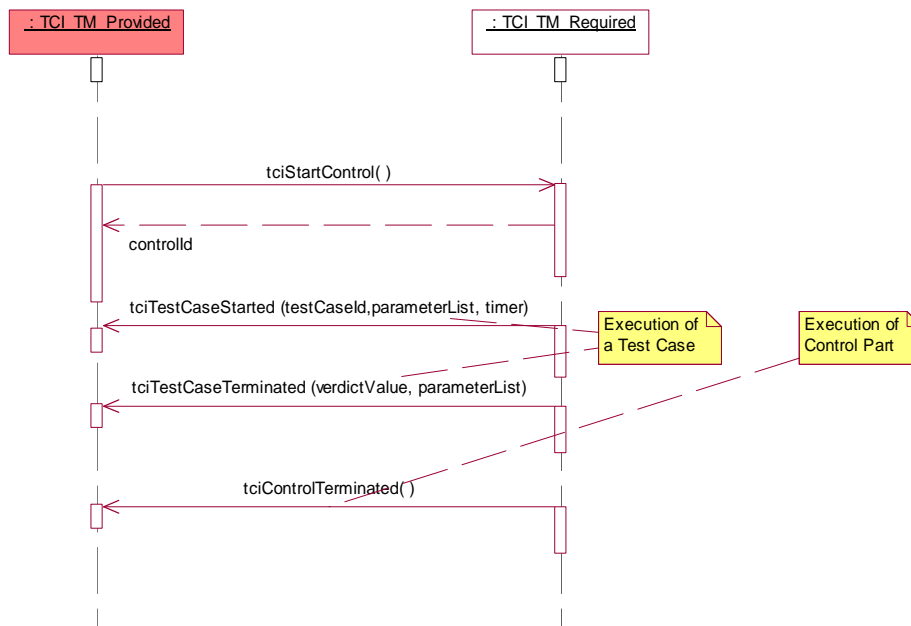


Figure 13: Use scenario - test case execution within control

12.2.2.2 TTCN-3 fragment

```

module AModule {
  ...
  testcase ATestCase(...)... {
    ... //the test case behaviour
  }
  ...
  control {
    ...
    execute (ATestCase (...));
    ...
  }
  ...
}

```

12.2.3 Use scenario: direct test case execution

The scenario in figure 14 shows how a test case can be directly executed from the test management outside the control part. After selecting the TTCN-3 module containing the test case to be executed, the start of the test case is requested. When the test case completes its execution, the test management is informed by the TE of the test case termination.

12.2.3.1 Sequence diagram

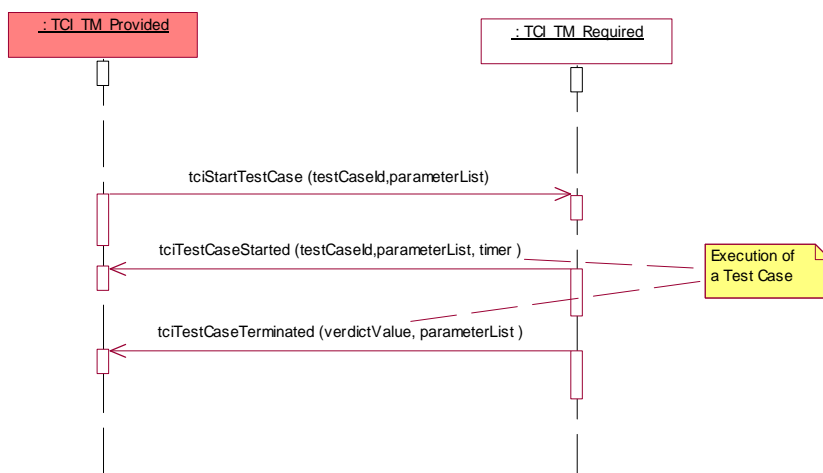


Figure 14: Use scenario - direct test case execution

12.2.3.2 TTCN-3 fragment

The direct execution of a test case is outside the scope of TTCN-3.

12.2.4 Use scenario: execute test case to TRI

The scenario in figure 15 shows how the TRI is informed about the execution of a test case so that it can set up and initialize system ports when needed. The execute test case request has to be issued before the test behaviour on the MTC of the current test case is started.

12.2.4.1 Sequence diagram

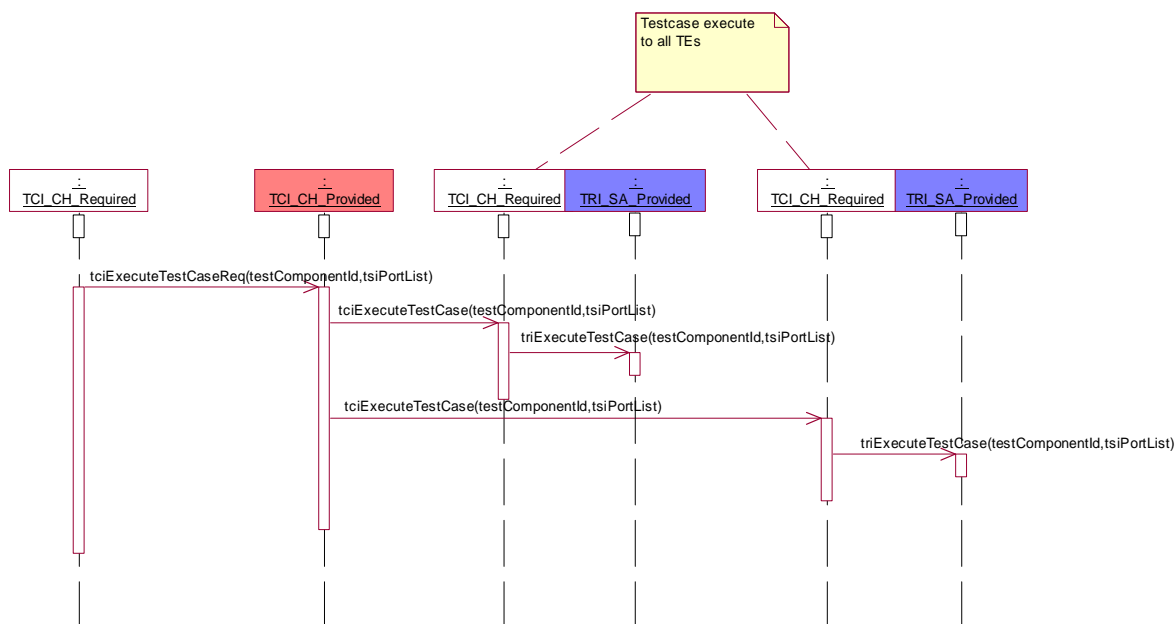


Figure 15: Use scenario - execute test case to TRI

12.2.4.2 TTCN-3 fragment

```

module AModule {
  ...
  testcase ATestCase(...)... {
    ... //the test case behaviour
  }
  ...
  control {
    ...
    execute(ATestCase (...));
    ...
  }
  ...
}

```

12.3 Component handling

12.3.1 Use scenario: local control component creation

The scenario in figure 16 demonstrates the creation of the control component on the same node where the user interface to the test management TCI-TM resides. A control component is created whenever the control part of a TTCN-3 module is executed. Whenever the test management TCI-TM issues the start of the control part, a create test component request is sent to the TCI-CH, which propagates it to the TE where the control component should be created. In this case it is the TE on the same node. The identifier for the control component is returned and given to the TCI-TM. The identifier is then used to start the behaviour of the control part on the control component.

12.3.1.1 Sequence diagram

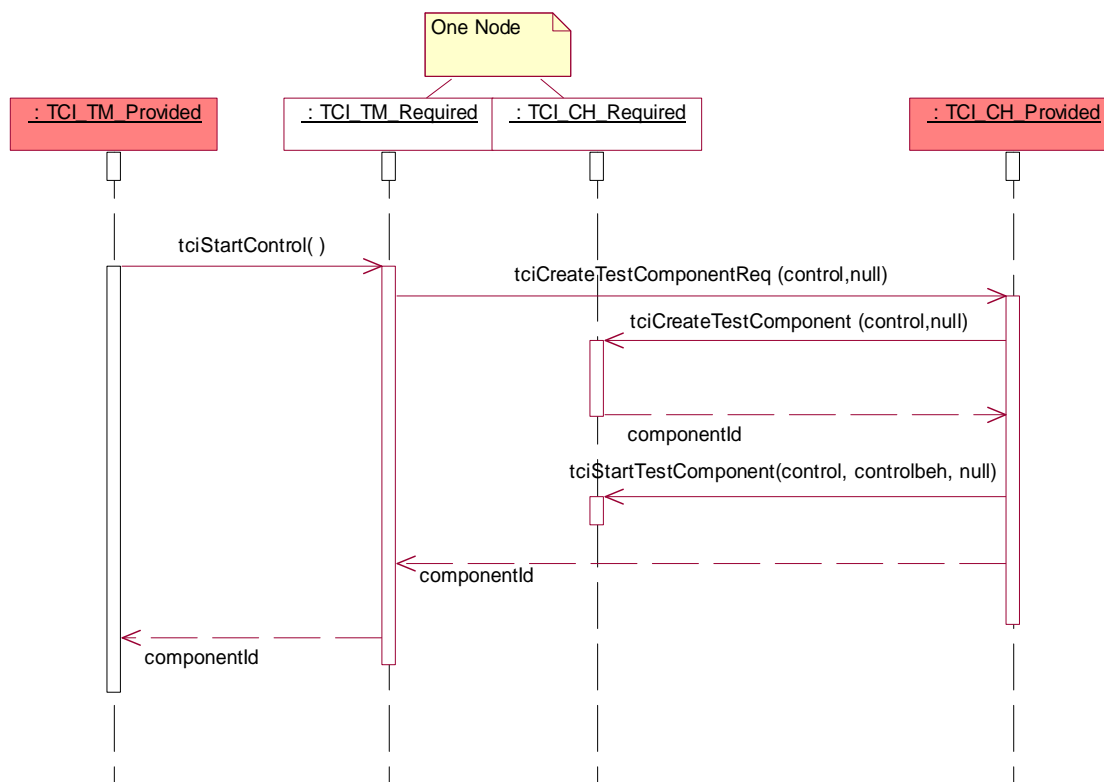


Figure 16: Use scenario - local control component creation

12.3.1.2 TTCN-3 fragment

```

module AModule {
  ...
  control {
    ...
  }
  ...
}

```

12.3.2 Use scenario: remote control component creation

The scenario in figure 17 demonstrates the creation of the control component on another node than where the user interface to the test management TCI-TM resides. A control component is created whenever the control part of a TTCN-3 module is executed. Whenever the test management TCI-TM issues the start of the control part, a create test component request is sent to the TCI-CH, which propagates it to the TE where the control component should be created. In this case it is the TE on another remote node. The identifier for the control component is returned and given to the TCI-TM. The identifier is then used to start the behaviour of the control part on the control component.

12.3.2.1 Sequence diagram

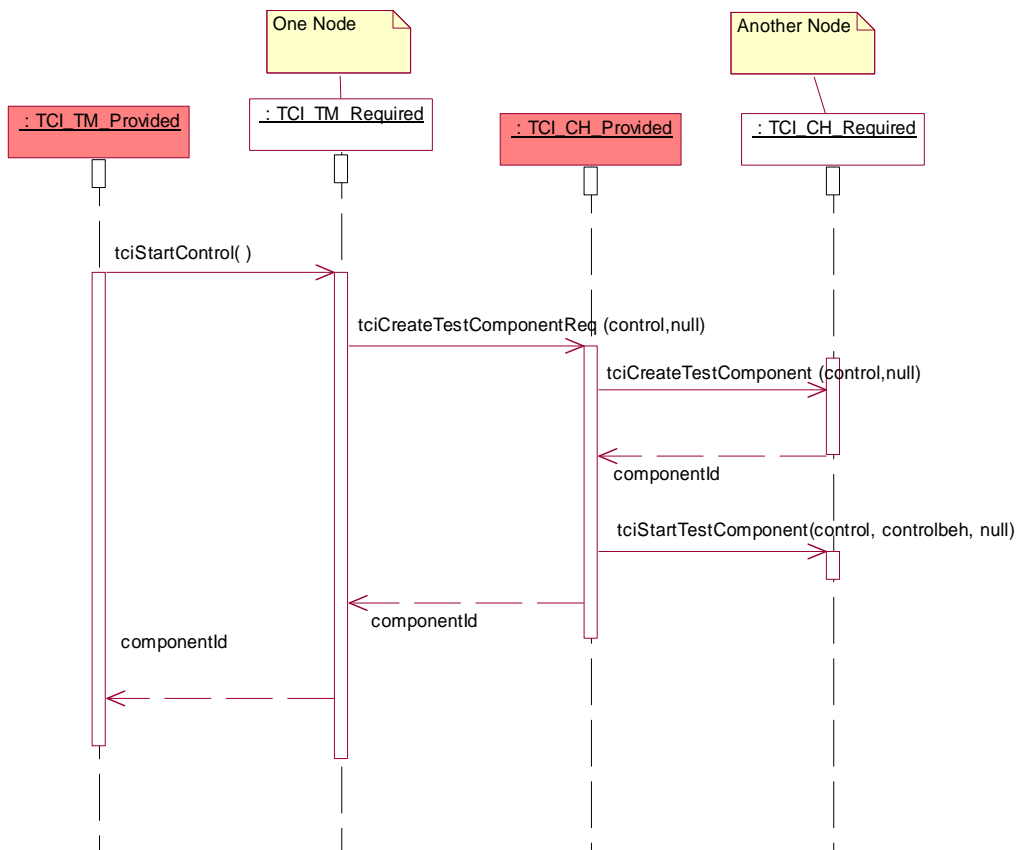


Figure 17: Use scenario - remote control component creation

12.3.2.2 TTCN-3 fragment

```

module AModule {
  ...
  control {
    ...
  }
  ...
}

```


12.3.3 Use scenario: local MTC creation

The scenario in figure 18 demonstrates the local creation of the main test component. Local is meant for two cases:

- 1) on the same node where the user interface to the test management TCI-TM resides (when a test case is started directly); or
- 2) on the same node where the control component resides (when a test case is executed from a control part).

A main test component is created whenever a test case is executed: a create test component request is sent to the TCI-CH, which propagates it to the TE where the main test component should be created. In this case it is the TE on the same node. The identifier for the main test component is returned and given to the TCI-TM. The identifier is then used to start the test case behaviour on the main test component (this is not shown here, but handled the same way as in the scenarios described in clauses 12.3.5 and 12.3.6).

12.3.3.1 Sequence diagram

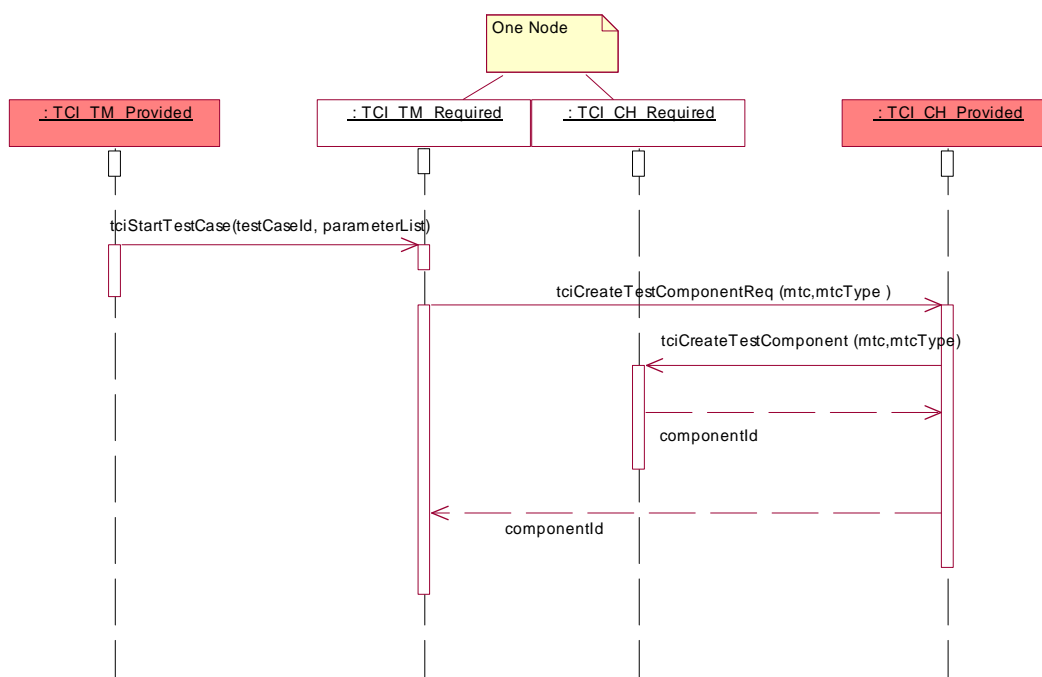


Figure 18: Use scenario - local MTC creation

12.3.3.2 TTCN-3 fragment

```

module AModule {
  ...
  testcase ATestCase (...) runs on MTCType... {
    ... //the test case behaviour
  }
  ...
}
  
```

12.3.4 Use scenario: remote MTC creation

The scenario in figure 19 demonstrates the remote creation of the main test component. Remote is meant for two cases:

- 1) on another node than where the user interface to the test management TCI-TM resides (when a test case is started directly); or
- 2) on another node than where the control component resides (when a test case is executed from a control part).

A main test component is created whenever a test case is executed: a create test component request is sent to the TCI-CH, which propagates it to the TE where the main test component should be created. In this case it is the TE on another node. The identifier for the main test component is returned and given to the TCI-TM. The identifier is then used to start the test case behaviour on the main test component (this is not shown here, but handled the same way as in the scenarios described in clauses 12.3.5 and 12.3.6).

12.3.4.1 Sequence diagram

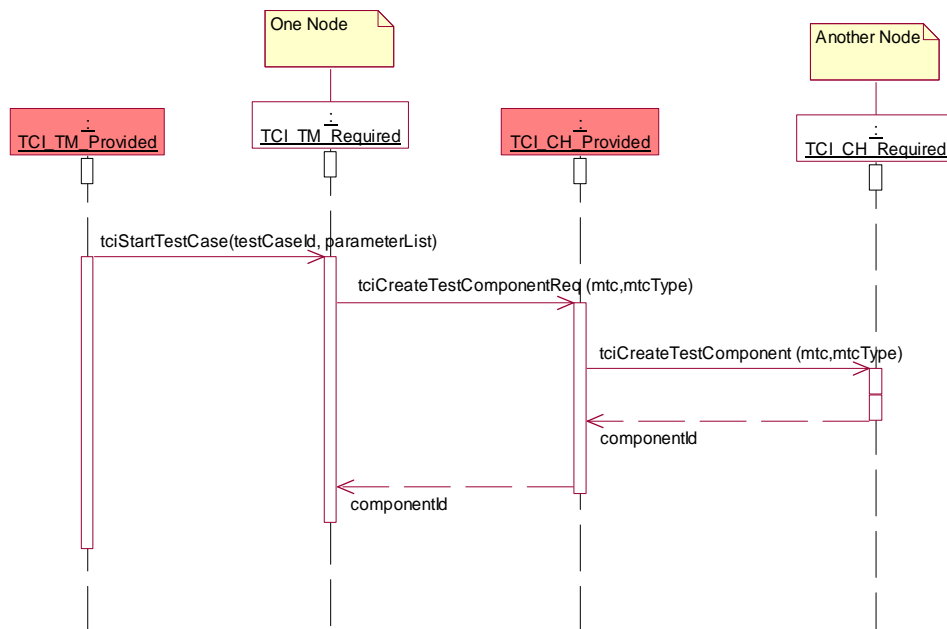


Figure 19: Use scenario - remote MTC creation

12.3.4.2 TTCN-3 fragment

```

module AModule {
  ...
  testcase ATestCase(...) runs on MTCType ... {
    ... //the test case behaviour
  }
  ...
}
  
```

12.3.5 Use scenario: component handling for test case execution within control

The scenario in figure 20 demonstrates the handling of components for the test case execution within a control part. When the control part is started, a control component is created and its component identifier returned to the test management. For each test case to be executed within the control part, a main test component is created and the component identifier returned to the control component. Afterwards, the test case behaviour is started on the main test component and the test management is informed about the start of the test case. When the main test component terminates, a request for the main test component termination together with the local verdict of the main test component is propagated to enable the derivation of the global test verdict and to enable the information about the test case termination.

12.3.5.1 Sequence diagram

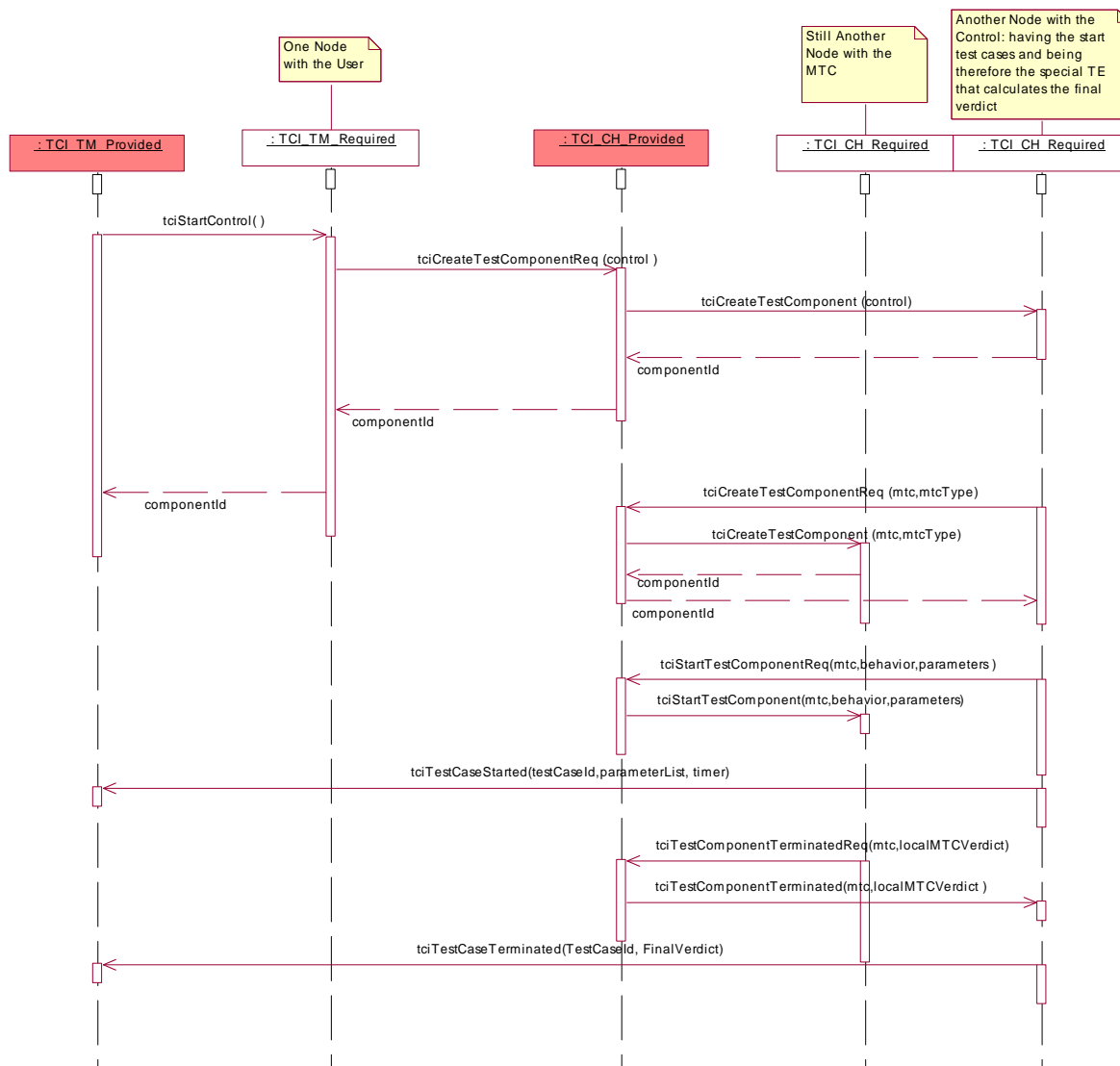


Figure 20: Use scenario: component handling for test case execution within control

12.3.5.2 TTCN-3 fragment

```

module AModule {
  ...
  testcase ATestCase(...)... {
    ... //the test case behaviour
  }
  ...
  control {
    ...
    execute (ATestCase (...)) ;
    ...
  }
  ...
}

```

12.3.6 Use scenario: component handling for direct test case execution

The scenario in figure 21 shows how test components are handled when a test case is executed directly, i.e. outside a control part. When a test case is started, the main test component is created and the test case behaviour started on this main test component at first. Whenever a parallel test component is used within a test case, it is handled the same: the parallel test component is started first: giving a test component create request to the TCI-CH entity, which propagates the test component create to the TE in which the parallel test component shall be created. The identifier for the created parallel test component is returned. The identifier is then used to start the PTC behaviour for the start operation. When the PTC terminates its execution, a test component terminate request together with the local test verdict is issued to inform TCI-CH about this termination. The same is done when the main test component terminates. In addition, the termination of the main test component leads to the overall termination of the test case.

12.3.6.1 Sequence diagram

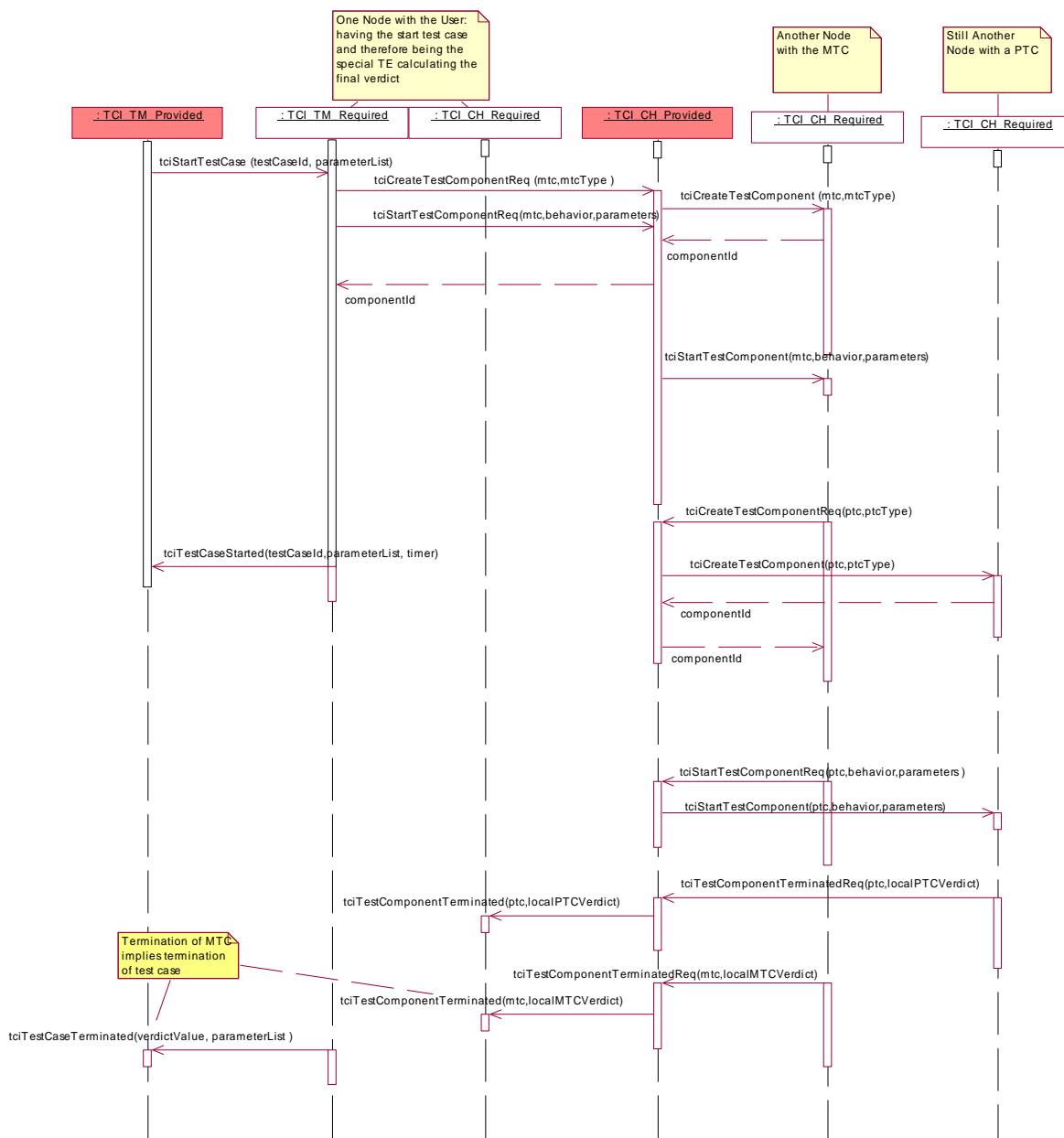


Figure 21: Use scenario: component handling for direct test case execution

12.3.6.2 TTCN-3 fragment

```

module AModule {
  ...
  function APTCBehaviour(...) runs on APTCType {
    ... //the PTC behaviour
  }
  ...
  testcase ATestCase(...)... {
    ... //the test case behaviour
    var APTCType PTC:= APTCType.create;
    ...
    PTC.start(APTCBehaviour(...));
    ...
  }
  ...
}

```

12.3.7 Use scenario: propagation of map/connect

The scenario in figure 22 shows how ports are mapped. The request to map a port is propagated to the TE where the map is finally performed. The propagation of connect requests works analogously.

12.3.7.1 Sequence diagram

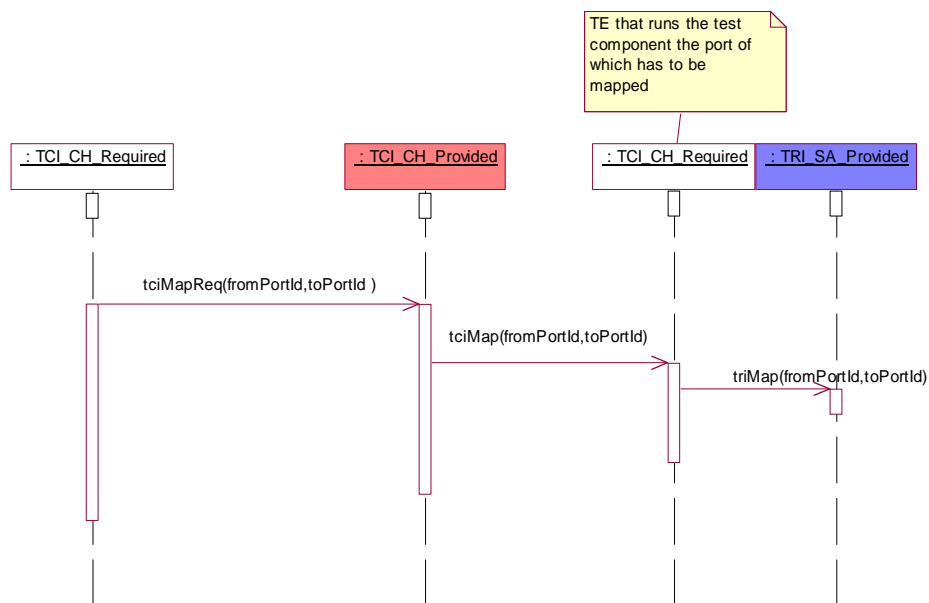


Figure 22: Use scenario: propagation of map

12.3.7.2 TTCN-3 fragment

```

module AModule {
  ...
  type port A { ... }
  type component CA { port A a }
  type component CB { port A a }
  ...
  testcase ATestCase(...)runs on CA system CB {
    var CA ptc := CA.create;
    ... //the test case behaviour
    map(ptc:a, System:a);
    ...
  }
  ...
}

```

12.3.8 Use scenario: propagation of unmap/disconnect

The scenario in figure 23 shows how ports are unmapped. The request to unmap a port is propagated to the TE where the unmap is finally performed. The propagation of disconnect requests works analogously.

12.3.8.1 Sequence diagram

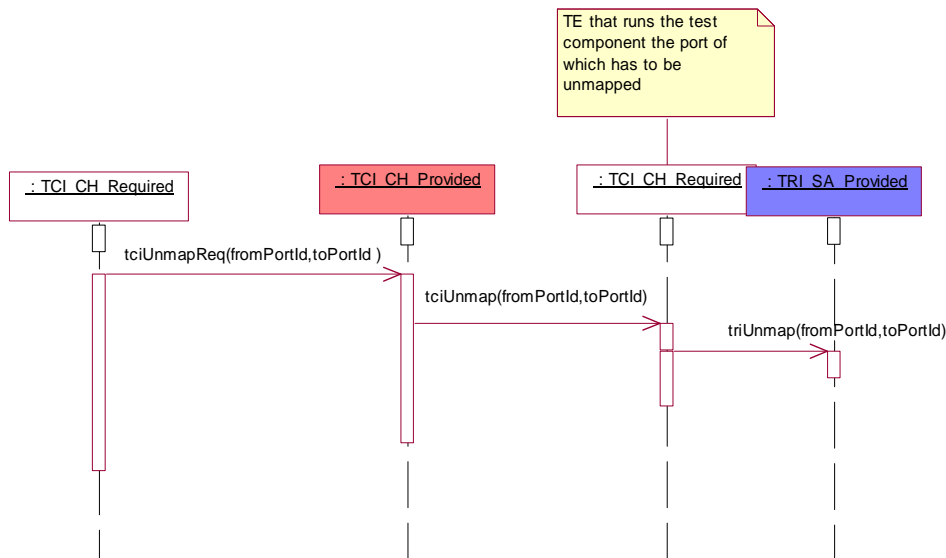


Figure 23: Use scenario - propagation of map

12.3.8.2 TTCN-3 fragment

```

module AModule {
  ...
  type port A { ... }
  type component CA { port A a }
  type component CB { port A a }
  ...
  testcase ATestCase(...) runs on CA system CB {
    var CA ptc := CA.create;
    ... //the test case behaviour
    unmap(ptc:a,system:a);
  }
  ...
}
  
```

12.4 Termination of test cases and control

12.4.1 Use scenario: stop a test case

The scenario in figure 24 shows how a test case is stopped from the test management during test case execution. Once the TM has received information about a started test case, a stop test case can be requested up until receiving the information that the test case has been terminated. Upon stopping a test case, all parallel test components will be stopped and the test system will be reset.

12.4.1.1 Sequence diagram

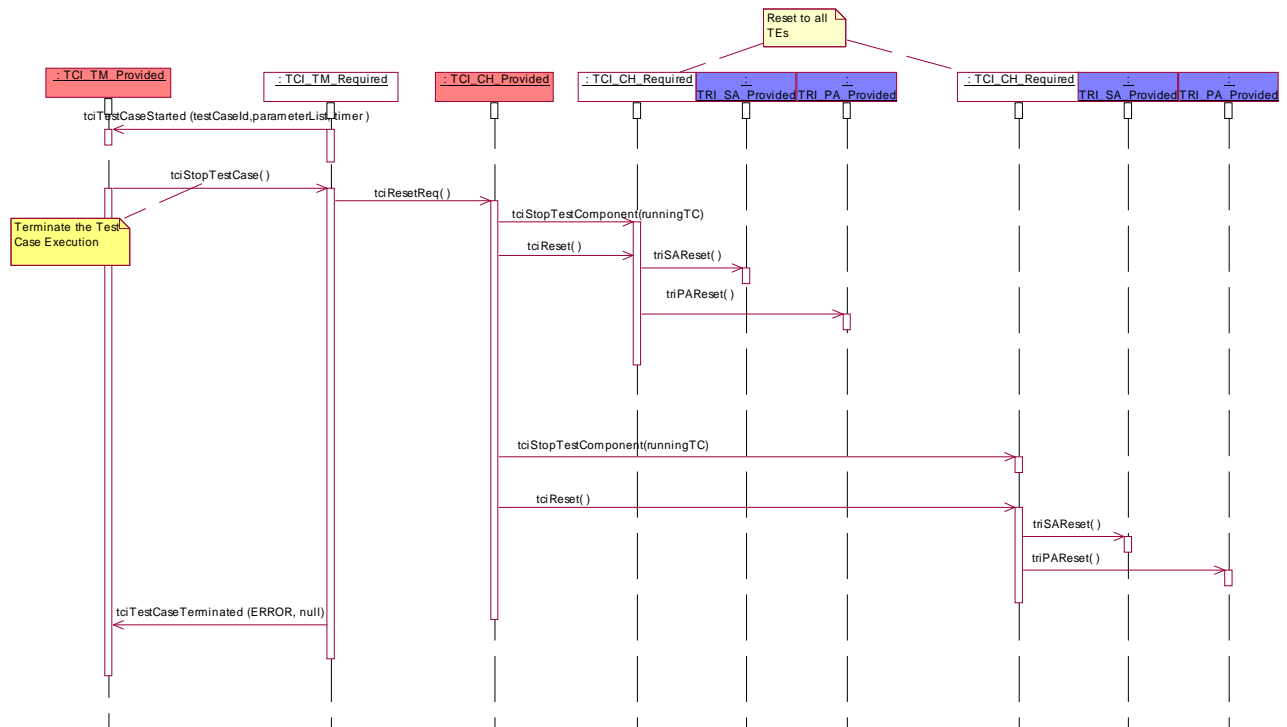


Figure 24: Use scenario: stop a test case

12.4.1.2 TTCN-3 fragment

There is no TTCN-3 code related to how the TM chooses to implement test case termination. This is outside the scope of TTCN-3.

12.4.2 Use scenario: stop control

The scenario in figure 25 shows how a control part is stopped from the test management during control part execution. A control part can be stopped in between starting the control and its termination. If the control part receives a stop test case request while a test case is executing, the executing test case shall be stopped. Furthermore, the test system shall be reset as described in figure 24.

12.4.2.1 Sequence diagram

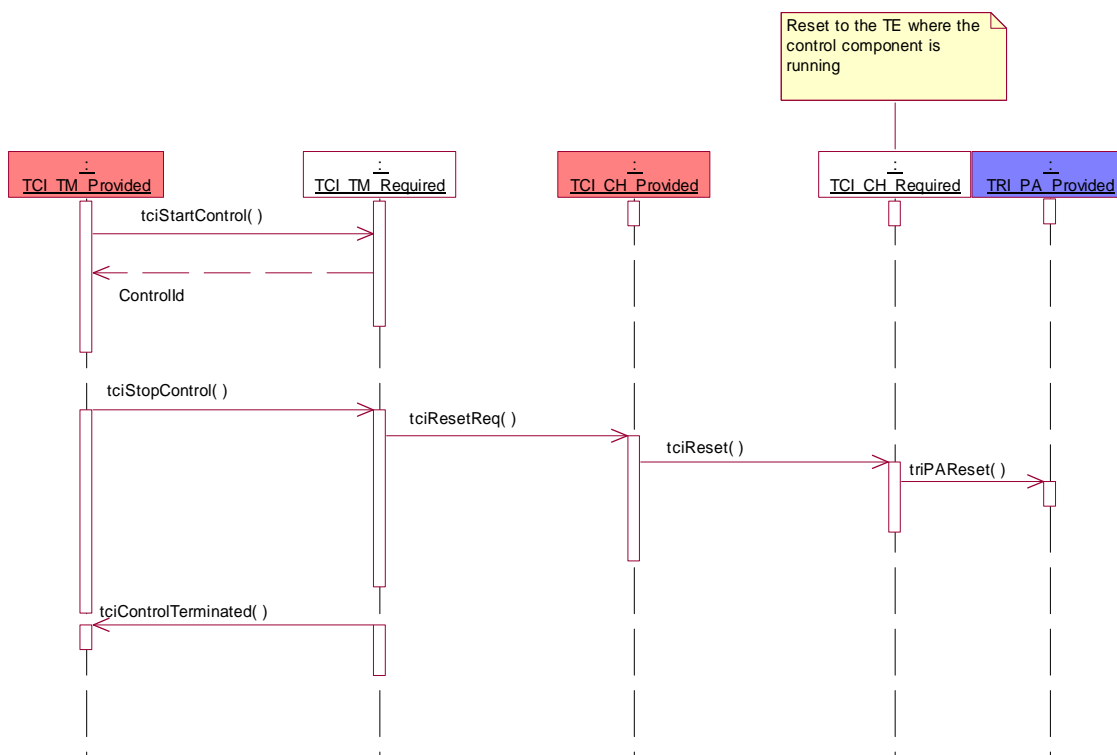


Figure 25: Use scenario - stop control

12.4.2.2 TTCN-3 fragment

Stopping a control part from the test management is outside the scope of TTCN-3 so that no TTCN-3 fragment exists.

12.4.3 Use scenario: termination of control after error

The scenario in figure 26 shows the handling of error situations during the execution of a control part when no test case is being executed. The test management is informed about the error situation and has then to terminate the execution of the control part explicitly. Upon termination of the control part, the test system will be reset.

12.4.3.1 Sequence diagram

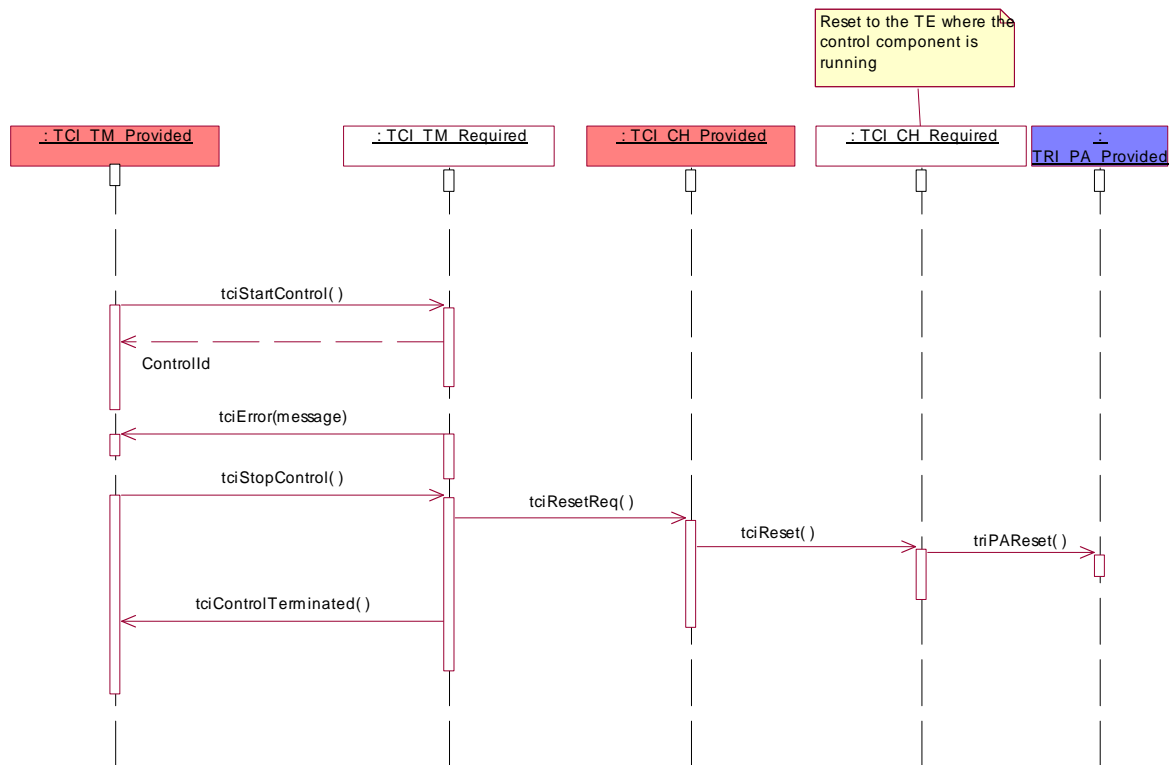


Figure 26: Use scenario - termination of control after error

12.4.3.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since error situations are exceptional cases in a test system and not a TTCN-3 concept as such. Rather, the TTCN-3 semantics describes various potential error situations in a test system.

12.4.4 Use scenario: termination of a test case after error

The scenario in figure 27 shows the handling of error situations during the direct execution of a test case. The test management is informed about the error situation. The TM must then explicitly terminate test case execution. Upon stopping a test case, the parallel test components will be stopped and the test system shall be reset.

12.4.4.1 Sequence diagram

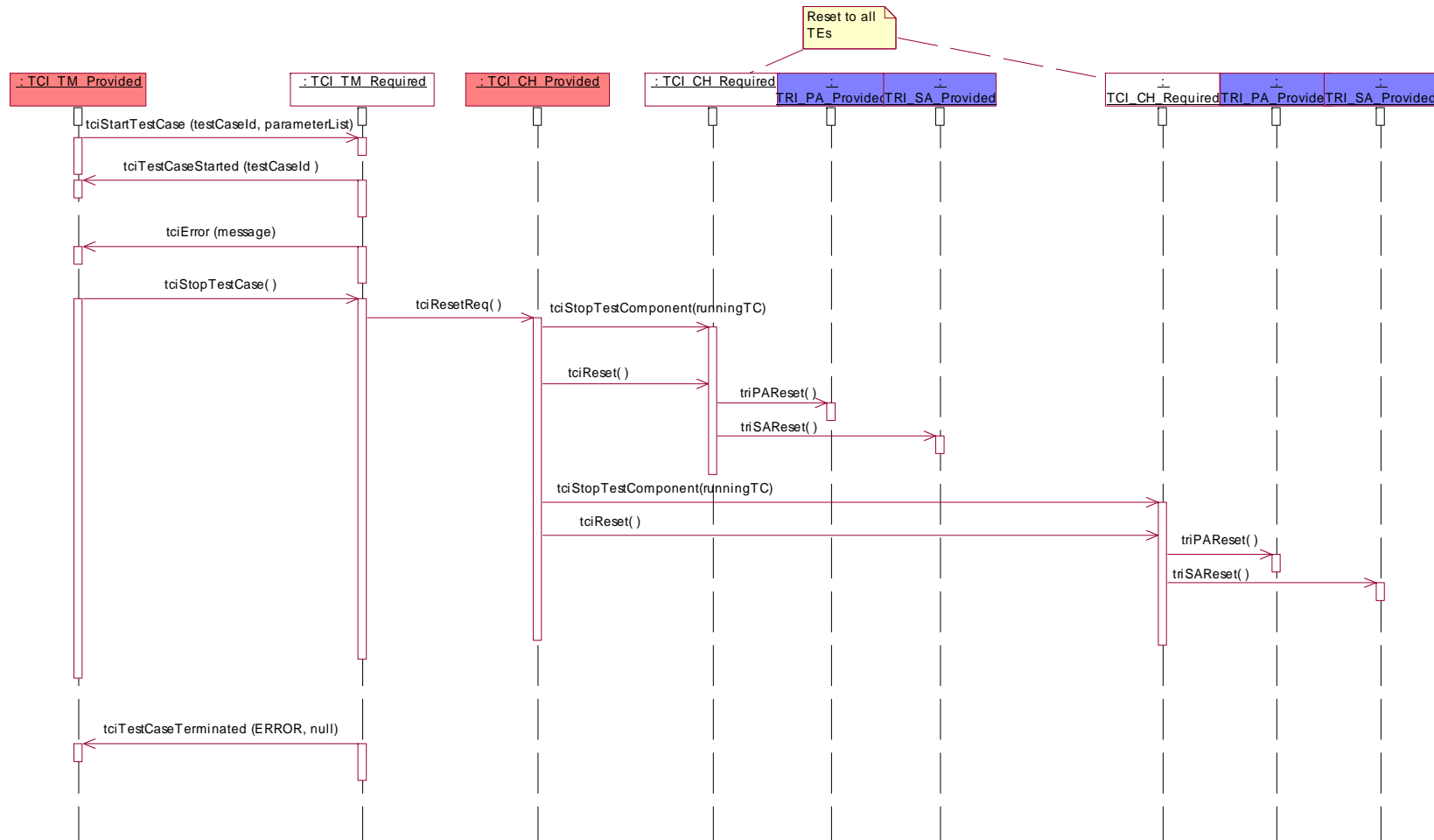


Figure 27: Use scenario - termination of a test case after error

12.4.4.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since error situations are exceptional cases in a test system and not a TTCN-3 concept as such. Rather, the TTCN-3 semantics describes various potential error situations in a test system.

12.4.5 Use scenario: reset

The scenario in figure 28 shows the reset of the test system. In that case all involved TEs together with their TRI System Adaptors (SA) and Platform Adaptors (PA) are reset.

12.4.5.1 Sequence diagram

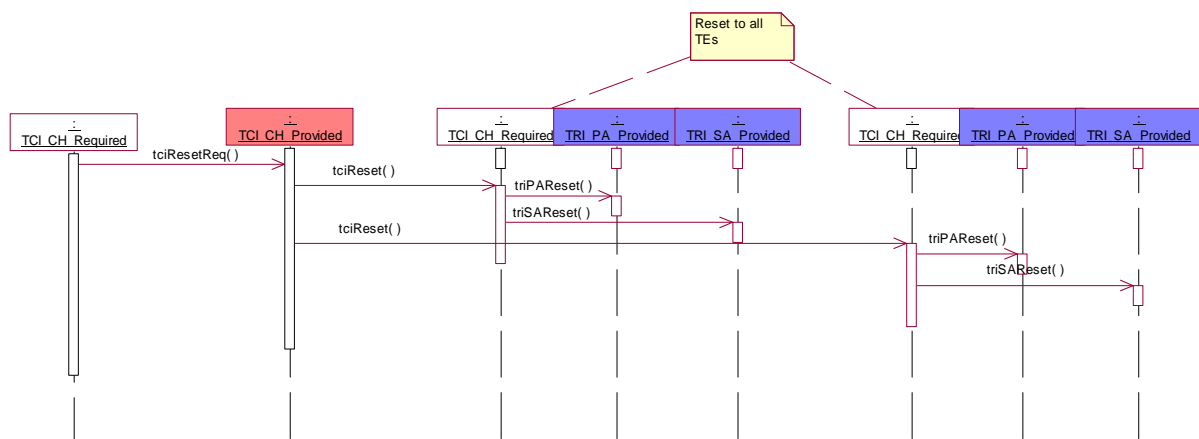


Figure 28: Use scenario - reset

12.4.5.2 TTCN-3 fragment

There is no TTCN-3 fragment for this scenario since reset as required after error situations are exceptional cases in a test system and not a TTCN-3 concept as such.

12.5 Communication

11.5.1 Use scenario: local intercomponent communication

The scenario in figure 29 shows the communication between test components (main test component or parallel test components), which reside on the same node. A communication request is given to the TCI-CH, which then decide where to enqueue this communication template. In this case, the communication is done locally via the TE on the same node. The scenario shows a message-based communication using the send operation - the scenario is the same for call, reply, and raise operations.

12.5.1.1 Sequence diagram

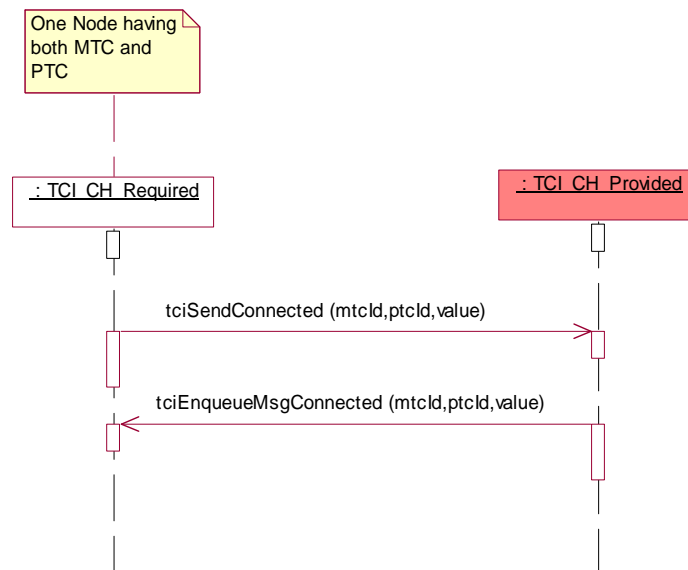


Figure 29: Use scenario - local intercomponent communication

12.5.1.2 TTCN-3 fragment

```

module AModule {
  ...
  type port APortType message { ... }
  ...
  type component ATCType {
    ...
    APortType APort;
    ...
  }
  ...
  template AType AMessageTemplate { ... }
  ...
  function APTCBehaviour(...) runs on APTCType {
    ... //the PTC behaviour
  }
  ...
  testcase ATestCase(...) runs on ATCType... {
    ... //the test case behaviour
    var ATCType PTC1:= ATCType.create;
    connect (PTC1:APort, mtc:APort);
    ...
    PTC1.start(APTCBehaviour(...));
    APort.send(AMessageTemplate); //sending data to a test component
    ...
  }
  ...
}
  
```

12.5.2 Use scenario: internode communication between test components

The scenario in figure 30 shows the communication between test components (main test component or parallel test components), which reside on different nodes. A communication request is given to the TCI-CH, which then decides where to enqueue this communication template. In this case, the communication is done remotely via the TE on another node. The scenario shows a message based communication using the send operation - the scenario is the same for call, reply, and raise operations.

12.5.2.1 Sequence diagram

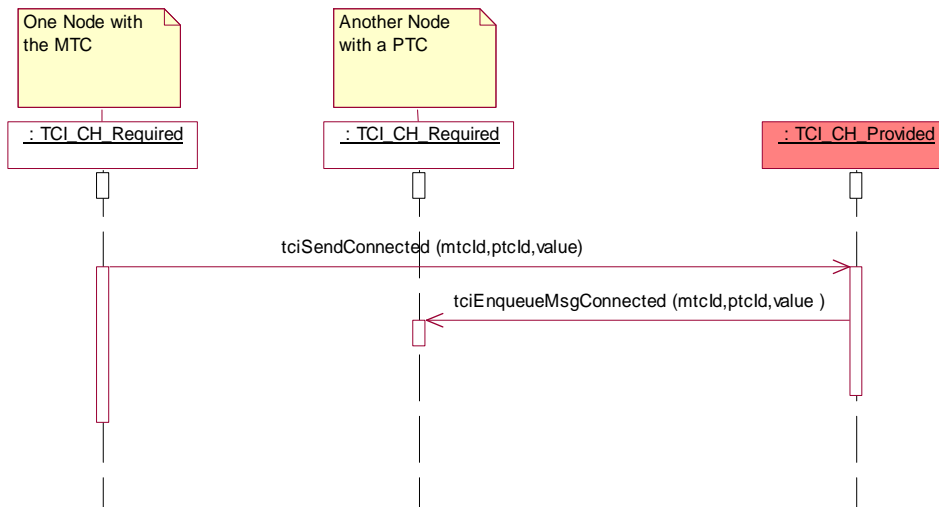


Figure 30: Use scenario - internode communication between test components

12.5.2.2 TTCN-3 fragment

```

module AModule {
  ...
  type port APortType message { ... }
  ...
  type component ATCType {
    ...
    APortType APort;
    ...
  }
  ...
  template AType AMessageTemplate { ... }
  ...
  function APTCBehaviour(...) runs on APTCType {
    ... //the PTC behaviour
  }
  ...
  testcase ATestCase(...) runs on ATCType... {
    ... //the test case behaviour
    var ATCType PTC1:= ATCType.create;
    connect (PTC1:APort,mtc:APort);
    ...
    PTC1.start(APTCBehaviour(...));
    APort.send(AMessageTemplate); //sending data to a test component
    ...
  }
  ...
}
  
```

12.5.3 Use scenario: encoding

The scenario in figure 31 shows the encoding of data, which is sent to the SUT. The encoded data is received from the coding/decoding entity via the TCI-CD. The encoded value is sent to the SUT via the TRI-SA. The scenario is the same for the call, the reply, and the raise operations.

12.5.3.1 Sequence diagram

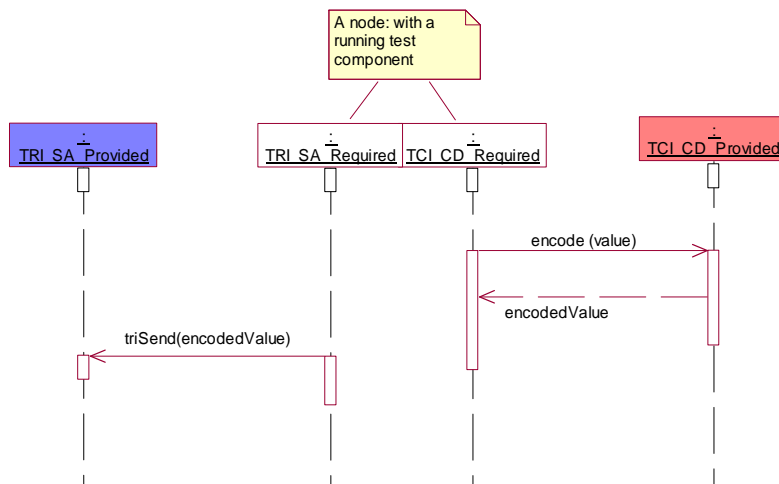


Figure 31: Use scenario - encoding

12.5.3.2 TTCN-3 fragment

```

module AModule {
  ...
  type port APortType message { ... }
  ...
  type component APTCType {
    ...
    APortType APort;
    ...
  }
  ...
  template AType AMessageTemplate { ... }
  ...
  testcase ATestCase(...) runs on APTCType system APTCType {
    ... //the test case behaviour
    map(mtc:APort, system:APort);
    ...
    APort.send(AMessageTemplate); //sending data to the SUT
    ...
  }
  ...
} with { encoding = '...' }
  
```

12.5.4 Use scenario: decoding

The scenario in figure 32 shows the decoding of data, which is received from the SUT via the TRI-SA. The decoded data is received from the coding/decoding entity via the TCI-CD. The scenario is the same for the receive, the getcall, the getreply, the catch, and the check operations.

12.5.4.1 Sequence diagram

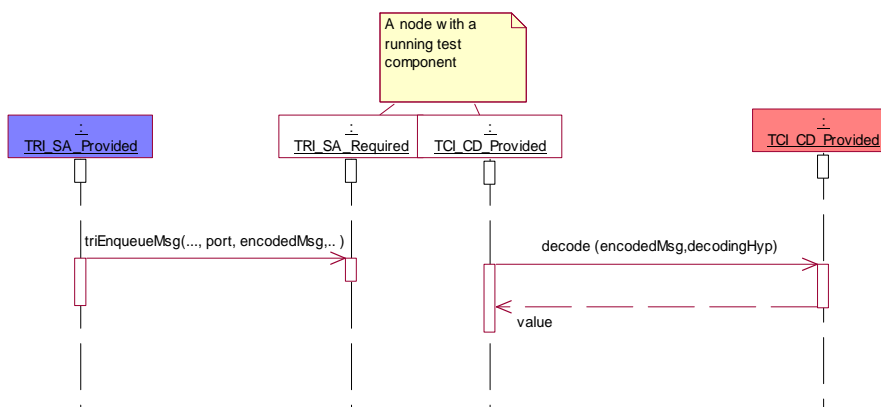


Figure 32: Use scenario - decoding

12.5.4.2 TTCN-3 fragment

```

module AModule {
  ...
  type port APortType message { ... }
  ...
  type component APTCType {
    ...
    APortType APort;
    ...
  }
  ...
  template AType AMessageTemplate { ... }
  ...
  testcase ATestCase(...) runs on APTCType system APTCType {
    ... //the test case behaviour
    map(mtc:APort, system:APort);
    ...
    APort.receive(AMessageTemplate); //receiving data from the SUT
    ...
  }
  ...
} with { encoding = '...' }

```

Annex A (normative): IDL Specification of TCI

This annex defines the TTCN-3 Control Interfaces using the Interface Definition Language (IDL).

```
// *****
// * Interface definitions for the TTCN-3 Control Interfaces
// *****

module tciInterface {

    /* Forward declaration */
    interface Value;
    interface Type;

    // *****
    // * Data types taken from the TRI definitions
    // *****

    // Connection
    native TriPortIdType ;
    native TriPortIdListType;
    native TriComponentIdType ;
    native TriComponentIdListType;

    // Communications
    native TriMessageType;
    native TriParameterType;
    native TriParameterListType;
    native TriAddressType;
    native TriAddressListType;
    native TriExceptionType;
    native TriSignatureIdType;

    // Miscellaneous
    native TriStatusType;
    native TriTimerIdType;
    native TriTimerDurationType;

    native TciStatusType;

    // *****
    // * General Abstract Data Types
    // *****

    // Basic definitions
    native TBoolean;
    native TFloat;
    native TChar;
    native TInteger;
    native TString;
    native TUniversalChar;
    typedef sequence <TString> TStringSeq;

    struct QualifiedName {
        TString moduleName;
        TString baseName;
    };

    // General TCI abstract data types
    typedef QualifiedName TciBehaviourIdType;
    typedef QualifiedName TciModuleIdType;
    typedef QualifiedName TciModuleParameterIdType;
    typedef QualifiedName TciTestCaseIdType;

    enum TciParameterPassingModeType {
        IN_MODE,
        OUT_MODE,
        INOUT_MODE
    };

    struct TciParameterType {
        TciModuleParameterIdType parameterName;
        Value parameterValue;
    };
};
```



```

    TciParameterPassingModeType mode;
};

typedef sequence <TciParameterType> TciParameterListType;

struct TciParameterTypeType {
    Type parameterType;
    TciParameterPassingModeType mode;
};

typedef sequence <TciParameterTypeType> TciParameterTypeListType;

struct TciModuleParameterType {
    TciModuleParameterIdType parameterName;
    Value defaultValue;
};

typedef sequence <TciModuleIdType> TciModuleIdListType ;

typedef sequence <TciModuleParameterType> TciModuleParameterListType;

typedef sequence <TciTestCaseIdType> TciTestCaseIdListType;

enum TciTestComponentKindType {
    CONTROL,
    MTC,
    PTC,
    SYSTEM,
    PTC_ALIVE
};

enum ComponentStatusType{
    inactiveC,
    runningC,
    stoppedC,
    killedC
};

enum TimerStatusType{
    runningT,
    inactiveT,
    expiredT
};

enum PortStatusType{
    startedP,
    haltedP,
    stoppedP
};

enum TciTypeClassType {
    ADDRESS_CLASS,
    ANYTYPE_CLASS,
    BITSTRING_CLASS,
    BOOLEAN_CLASS,
    CHAR_CLASS,
    CHARSTRING_CLASS,
    COMPONENT_CLASS,
    ENUMERATED_CLASS,
    FLOAT_CLASS,
    HEXSTRING_CLASS,
    INTEGER_CLASS,
    OCTETSTRING_CLASS,
    RECORD_CLASS,
    RECORDOF_CLASS,
    ARRAY_CLASS,
    SET_CLASS,
    SETOF_CLASS,
    UNION_CLASS,
    UNIVERSALCHAR_CLASS,
    UNIVERSALCHARSTRING_CLASS,
    VERDICT_CLASS
};

```

```

// *****
// * Abstract TTCN-3 Data Types And Values
// *****

// Abstract data type "Type"
interface Type {
  TciModuleIdType  getDefiningModule ();
  TString          getName ();
  TciTypeClassType getTypeClass ();
  Value           newInstance ();
  TString         getTypeEncoding ();
  TString         getTypeEncodingVariant ();
  TStringSeq      getTypeextension ();
};

// Abstract TTCN-3 Values
interface Value {
  TString  getValueEncoding ();
  TString  getValueEncodingVariant ();
  Type     getType ();
  TBoolean notPresent ();
};

interface RecordOfValue : Value {
  Value  getField (in TInteger position);
  void   setField (
    in TInteger position,
    in Value value
  );
  void   appendField (in Value value);
  Type   getElementType ();
  TInteger getLength ();
  void   setLength (in TInteger len);
  TInteger getOffset ();
};

interface RecordValue : Value {
  Value  getField (in TString fieldName);
  void   setField (
    in TString fieldName,
    in Value value
  );
  TStringSeq getFieldNames ();
  void   setFieldOmitted (in TString fieldName);
};

interface VerdictValue : Value {
  TInteger getVerdict ();
  void   setVerdict (in TInteger verdict);
};

interface BitstringValue : Value {
  TString  getString ();
  void   setString (in TString value);
  TInteger getBit (in TInteger position);
  void   setBit (
    in TInteger position,
    in TInteger value
  );
  TInteger getLength ();
  void   setLength (in TInteger len);
};

interface OctetstringValue : Value {
  TString  getString ();
  void   setString (in TString value);
  TInteger getOctet (in TInteger position);
  void   setOctet (
    in TInteger position,
    in TInteger value
  );
  TInteger getLength ();
  void   setLength (in TInteger len);
};

interface FloatValue : Value {
  TFloat getFloat ();
  void   setFloat (in TFloat value);
};

```

```

};

interface HexstringValue : Value {
    TString getString ();
    void setString (in TString value);
    TInteger getHex (in TInteger position);
    void setHex (
        in TInteger position,
        in TInteger value
    );
    TInteger getLength ();
    void setLength (in TInteger len);
};

interface EnumeratedValue : Value {
    void setEnum (in TString enumValue);
    TString getEnum ();
};

interface IntegerValue : Value {
    TInteger getInt ();
    void setInt (in TInteger value);
};

interface CharValue : Value {
    TChar getChar ();
    void setChar (in TChar value);
};

interface CharstringValue : Value {
    TString getString ();
    void setString (in TString value);
    TChar getChar (in TInteger position);
    void setChar (
        in TInteger position,
        in TChar value
    );
    TInteger getLength ();
    void setLength (in TInteger len);
};

interface BooleanValue : Value {
    TBoolean getBoolean ();
    void setBoolean (in TBoolean value);
};

interface UniversalCharValue : Value {
    TUniversalChar getUniversalChar ();
    void setUniversalChar (in TUniversalChar value);
};

interface UniversalCharstringValue : Value {
    TString getString ();
    void setString (in TString value);
    TUniversalChar getChar (in TInteger position);
    void setChar (
        in TInteger position,
        in TUniversalChar value
    );
    TInteger getLength ();
    void setLength (in TInteger len);
};

interface UnionValue : Value {
    Value getVariant (in TString variantName);
    void setVariant (
        in TString variantName,
        in Value value
    );
    TString getPresentVariantName ();
    TStringSeq getVariantNames ();
};

```

```

// *****
// * Abstract Logging Types
// *****

interface TciValueTemplate : Value {
    TBoolean isOmit ();
    TBoolean isAny ();
    TBoolean isAnyOrOmit ();
    TString getTemplateDef ();
};

interface TciNonValueTemplate {
    TBoolean isAny ();
    TBoolean isAll ();
    TString getTemplateDef ();
};

typedef sequence <Value> TciValueListType;

struct TciValueDifferenceType
{
    TString desc;
    Value val;
    TciValueTemplate tmpl;
};

typedef sequence <TciValueDifferenceType> TciValueDifferenceListType;

interface TciValueList {
    attribute TciValueListType inst;
    TInteger size ();
    TBoolean isEmpty ();
    Value get (in TInteger index);
};

interface TciValueDifference {
    attribute TciValueDifferenceType inst;
    Value getValue ();
    TciValueTemplate getTciValueTemplate ();
    TString getDescription ();
};

interface TciValueDifferenceList {
    attribute TciValueDifferenceListType inst;
    TInteger size ();
    TBoolean isEmpty ();
    TciValueDifference get (in TInteger index);
};

// *****
// Coding Decoding Interface
// - Required
// *****

interface TCI_CD_Required {
    Type getTypeForName (in TString typeName);
    Type getInteger ();
    Type getFloat ();
    Type getBoolean ();
    Type getChar ();
    Type getUniversalChar ();
    Type getCharstring ();
    Type getUniversalCharstring ();
    Type getHexstring ();
    Type getBitstring ();
    Type getOctetstring ();
    Type getVerdict ();
    void tciErrorReq (in TString message);
};

// *****
// Coding Decoding interface
// - Provided
// *****

interface TCI_CD_Provided {
    Value decode (
        in TriMessageType message,

```

```

        in Type decodingHypothesis
    );
    TriMessageType encode (in Value value);
};

// *****
// Test Management Interface
// - Required
// *****

interface TCI_TM_Required : TCI_CD_Required {
    void tciRootModule (in TciModuleIdType moduleName);
    TciModuleIdListType tciGetImportedModules();
    TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName);
    TciTestCaseIdListType tciGetTestCases ();
    TciParameterTypeListType tciGetTestCaseParameters (
        in TciTestCaseIdType testCaseId
    );
    TriPortIdListType tciGetTestCaseTSI (
        in TciTestCaseIdType testCaseId
    );
    void tciStartTestCase (
        in TciTestCaseIdType testCaseId,
        in TciParameterListType parameterList
    );
    void tciStopTestCase ();
    TriComponentIdType tciStartControl ();
    void tciStopControl ();
};

// *****
// Test Management Interface
// - Provided
// *****

interface TCI_TM_Provided {
    void tciTestCaseStarted (
        in TciTestCaseIdType testCaseId,
        in TciParameterListType parameterList,
        in TFloat timer
    );
    void tciTestCaseTerminated (
        in VerdictValue verdict,
        in TciParameterListType parameterList
    );
    void tciControlTerminated ();
    Value tciGetModulePar (
        in TciModuleParameterIdType parameterId
    );
    void tciLog (
        in TriComponentIdType testComponentId,
        in TString message
    );
    void tciError (in TString message);
};

// *****
// Component Handling Interface
// - Required
// *****

interface TCI_CH_Required : TCI_CD_Required {
    void tciEnqueueMsgConnected (
        in TriPortIdType sender,
        in TriComponentIdType receiver,
        in Value receivedMessage
    );
    void tciEnqueueCallConnected (
        in TriPortIdType sender,
        in TriComponentIdType receiver,
        in TriSignatureIdType signature,
        in TciParameterListType parameterList
    );
    void tciEnqueueReplyConnected (
        in TriPortIdType sender,
        in TriComponentIdType receiver,

```

```

        in TriSignatureIdType signature,
        in TciParameterListType parameterList,
        in Value returnValue
    );
void tciEnqueueRaiseConnected (
    in TriPortIdType sender,
    in TriComponentIdType receiver,
    in TriSignatureIdType signature,
    in Value except
);
TriComponentIdType tciCreateTestComponent (
    in TciTestComponentKindType kind,
    in Type componentType,
    in TString name
);
void tciStartTestComponent (
    in TriComponentIdType comp,
    in TciBehaviourIdType behavior,
    in TciParameterListType parameterList
);
void tciStopTestComponent (
    in TriComponentIdType comp
);
void tciConnect (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciDisconnect (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciTestComponentTerminated (
    in TriComponentIdType comp,
    in VerdictValue verdict
);
TBoolean tciTestComponentRunning (
    in TriComponentIdType comp
);
TriComponentIdType tciGetMTC ();
void tciMap (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciUnmap (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciExecuteTestCase (
    in TciTestCaseIdType testCaseId,
    in TriPortIdListType tsiPortList
);
TBoolean tciTestComponentDone (
    in TriComponentIdType comp
);
void tciReset ();
};

// *****
// Component Handling Interface
// - Provided
// *****

interface TCI_CH_Provided {
    void tciSendConnected (
        in TriPortIdType sender,
        in TriComponentIdType receiver,
        in Value sendMessage
    );
    void tciSendConnectedBC (
        in TriPortIdType sender,
        in Value sendMessage
    );
    void tciSendConnectedMC (
        in TriPortIdType sender,
        in TriComponentIdListType receivers,
        in Value sendMessage
    );
};

```

```

void tciCallConnected (
    in TriPortIdType sender,
    in TriComponentIdType receiver,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList
);
void tciCallConnectedBC (
    in TriPortIdType sender,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList
);
void tciCallConnectedMC (
    in TriPortIdType sender,
    in TriComponentIdListType receivers,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList
);

void tciReplyConnected (
    in TriPortIdType sender,
    in TriComponentIdType receiver,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList,
    in Value returnValue
);
void tciReplyConnectedBC (
    in TriPortIdType sender,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList,
    in Value returnValue
);
void tciReplyConnectedMC (
    in TriPortIdType sender,
    in TriComponentIdListType receivers,
    in TriSignatureIdType signature,
    in TciParameterListType parameterList,
    in Value returnValue
);

void tciRaiseConnected (
    in TriPortIdType sender,
    in TriComponentIdType receiver,
    in TriSignatureIdType signature,
    in Value except
);
void tciRaiseConnectedBC (
    in TriPortIdType sender,
    in TriSignatureIdType signature,
    in Value except
);
void tciRaiseConnectedMC (
    in TriPortIdType sender,
    in TriComponentIdListType receivers,
    in TriSignatureIdType signature,
    in Value except
);

TriComponentIdType tciCreateTestComponentReq (
    in TciTestComponentKindType kind,
    in Type componentType,
    in TString name
);
void tciStartTestComponentReq (
    in TriComponentIdType comp,
    in TciBehaviourIdType behavior,
    in TciParameterListType parameterList
);
void tciStopTestComponentReq (
    in TriComponentIdType comp
);
void tciConnectReq (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciDisconnectReq (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);

```

```

void tciTestComponentTerminatedReq (
    in TriComponentIdType comp,
    in VerdictValue verdict
);
TBoolean tciTestComponentRunningReq (
    in TriComponentIdType comp
);
TriComponentIdType tciGetMTCReq ();
void tciMapReq (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciUnmapReq (
    in TriPortIdType fromPort,
    in TriPortIdType toPort
);
void tciExecuteTestCaseReq (
    in TciTestCaseIdType testCaseId,
    in TriPortIdListType tsiPortList
);
void tciResetReq ();
TBoolean tciTestComponentDoneReq (
    in TriComponentIdType comp
);
};

// *****
// Test Logging Interface
// - Provided
// *****

interface TCI_TL_Provided {
void tliTcExecute(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciTestCaseIdType tcId,
    in TciParameterListType tciPars, in TriTimerDurationType dur
);
void tliTcStart(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciTestCaseIdType tcId,
    in TciParameterListType tciPars, in TriTimerDurationType dur
);
void tliTcStop(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c
);
void tliTcStarted(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciTestCaseIdType tcId,
    in TciParameterListType tciPars, in TriTimerDurationType dur
);
void tliTcTerminated(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciTestCaseIdType tcId,
    in TciParameterListType tciPars, in VerdictValue verdict);

void tliCtrlStart(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c
);
void tliCtrlStop(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c
);
void tliCtrlTerminated(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c);

void tliMSend_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
    in Value addrValue, in TciStatusType encoderFailure,
    in TriMessageType msg, in TriAddressType address, in TriStatusType transmissionFailure
);
void tliMSend_m_BC(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to, in Value msgValue,
    in TciStatusType encoderFailure, in TriMessageType msg,

```



```

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

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

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

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

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

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

void tliMMismatch_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
    in TciValueTemplate msgTpl, in TciValueDifferenceList diffs,
    in Value addrValue, in TciValueTemplate addressTpl
);

void tliMMismatch_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
    in TciValueTemplate msgTpl, in TciValueDifferenceList diffs,
    in TriComponentIdType from, in TciNonValueTemplate fromTpl
);

void tliMReceive_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
    in TciValueTemplate msgTpl, in Value addrValue,
    in TciValueTemplate addressTpl
);

void tliMReceive_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in Value msgValue,
    in TciValueTemplate msgTpl, in TriComponentIdType from,
    in TciNonValueTemplate fromTpl
);

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

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

void tliPrCall_m_MC(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,

```

```

    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in TciValueList addrValues, in TciStatusType encoderFailure,
    in TriParameterListType triPars, in TriAddressListType addresses,
    in TriStatusType transmissionFailure
    );

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

void tliPrGetCallDetected_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
    in TriSignatureIdType signature, in TriParameterListType triPars,
    in TriAddressType address
    );
void tliPrGetCallDetected_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
    in TriSignatureIdType signature, in TciParameterListType tciPars
    );
void tliPrGetCallMismatch_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in TciValueTemplate parsTpl, in TciValueDifferenceList diffs,
    in Value addrValue, in TciValueTemplate addressTpl
    );
void tliPrGetCallMismatch_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in TciValueTemplate parsTpl, in TciValueDifferenceList diffs,
    in TriComponentIdType from, in TciNonValueTemplate fromTpl
    );
void tliPrGetCall_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in TciValueTemplate parsTpl, in Value addrValue,
    in TciValueTemplate addressTpl
    );
void tliPrGetCall_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in TciValueTemplate parsTpl, in TriComponentIdType from,
    in TciNonValueTemplate fromTpl
    );

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

```

```

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

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

void tliPrReply_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType to,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in Value replValue, in TriStatusType transmissionFailure
);

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

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

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

void tliPrGetReplyDetected_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at, in TriPortIdType from,
    in TriSignatureIdType signature, in TciParameterListType tciPars,
    in Value replValue
);

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

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

void tliPrGetReply_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature,
    in TciParameterListType tciPars, in TciValueTemplate parsTmpl,
    in Value replValue, in TciValueTemplate replyTmpl,
    in Value addrValue, in TciValueTemplate addressTmpl
);

```

```

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

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

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

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

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

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

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

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

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

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

void tliPrCatchMismatch_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature,
    in Value excValue, in TciValueTemplate excTmpl,
    in TciValueDifferenceList diffs, in TriComponentIdType from,
    in TciNonValueTemplate fromTmpl
);

```

```

void tliPrCatch_m(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature,
    in Value excValue, in TciValueTemplate excTmpl,
    in Value addrValue, in TciValueTemplate addressTmpl);

void tliPrCatch_c(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature,
    in Value excValue, in TciValueTemplate excTmpl,
    in TriComponentIdType from, in TciNonValueTemplate fromTmpl
);
void tliPrCatchTimeoutDetected(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature
);
void tliPrCatchTimeout(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType at,
    in TriSignatureIdType signature
);
void tliCCreate(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp,
    in TString name, in TBoolean alive
);
void tliCStart(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp,
    in TciBehaviourIdType name, in TciParameterListType tciPars
);
void tliCRunning(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp, in ComponentStatusType status
);
void tliCAlive(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c,
    in TriComponentIdType comp, in ComponentStatusType status
);
void tliCStop(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp
);
void tliCKill(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp
);
void tliCDoneMismatch(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl
);
void tliCKilledMismatch(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl
);
void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciNonValueTemplate compTmpl
);
void tliCKilled(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TciNonValueTemplate compTmpl
);
void tliCTerminated(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in VerdictValue verdict
);
void tliPConnect(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2
);
void tliPDisconnect(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port1,
    in TriPortIdType port2
);

```

```

    );
void tliPMap(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port1, in TriPortIdType port2
);
void tliPUnmap(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port1,
    in TriPortIdType port2
);
void tliPClear(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port
);
void tliPStart(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port
);
void tliPStop(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port
);
void tliPHalt(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriPortIdType port
);
void tliEncode(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in Value val, in TciStatusType encoderFailure,
    in TriMessageType msg, in TString codec
);
void tliDecode(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriMessageType msg,
    in TciStatusType decoderFailure, in Value val, in TString codec
);
void tliTimeoutDetected(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer
);
void tliTimeoutMismatch(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer, in TciNonValueTemplate timerTpl
);
void tliTimeout(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer, in TciNonValueTemplate timerTpl
);
void tliTStart(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer,
    in TriTimerDurationType dur
);
void tliTStop(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur
);
void tliTRead(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer,
    in TriTimerDurationType elapsed
);
void tliTRunning(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in TriTimerIdType timer, in TimerStatusType status
);
void tliSEnter(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars,
    in TString kind
);
void tliSLeave(
    in TString am, in TInteger ts, in TString src, in TInteger line,
    in TriComponentIdType c, in QualifiedName name, in TciParameterListType tciPars,
    in Value returnValue, in TString kind
);
void tliVar(
    in TString am, in TInteger ts, in TString src, in TInteger line,

```


Annex B (normative): XML Mapping for TCI TL Provided

This annex defines a mapping for the logging interface of TCI using eXtended Markup Language (XML) schema definitions.

B.1 TCI-TL XML Schema for Simple Types

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  elementFormDefault="qualified">

  <!-- Basic definitions -->
  <xsd:simpleType name="xpath">
    <!-- this string should be XPATH compliant -->
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:simpleType name="TBoolean">
    <xsd:restriction base="xsd:boolean"/>
  </xsd:simpleType>

  <xsd:simpleType name="TString">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:simpleType name="TInteger">
    <xsd:restriction base="xsd:integer"/>
  </xsd:simpleType>

  <!-- Miscellaneous -->
  <xsd:simpleType name="TriTimerDurationType">
    <xsd:restriction base="xsd:float"/>
  </xsd:simpleType>

  <xsd:simpleType name="TciParameterPassingModeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="in"/>
      <xsd:enumeration value="inout"/>
      <xsd:enumeration value="out"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="TriStatusType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="TRI_Ok"/>
      <xsd:enumeration value="TRI_Error"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="TciStatusType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="TCI_Ok"/>
      <xsd:enumeration value="TCI_Error"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="ComponentStatusType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="inactiveC"/>
      <xsd:enumeration value="runningC"/>
      <xsd:enumeration value="stoppedC"/>
      <xsd:enumeration value="killedC"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```



```

<xsd:simpleType name="TimerStatusType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="runningT"/>
    <xsd:enumeration value="inactiveT"/>
    <xsd:enumeration value="expiredT"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="PortStatusType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="startedP"/>
    <xsd:enumeration value="haltedP"/>
    <xsd:enumeration value="stoppedP"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

B.2 TCI-TL XML Schema for Types

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
  xmlns:Types="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
  xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  xmlns:Values="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  xmlns:Templates="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
  elementFormDefault="qualified">

  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
    schemaLocation="Values_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
    schemaLocation="SimpleTypes_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
    schemaLocation="Templates_v3_3_1.xsd"/>

  <!-- Connection -->
  <xsd:complexType name="TriPortIdType">
    <xsd:sequence>
      <xsd:element name="comp" type="Types:TriComponentIdType" />
      <xsd:element name="port" type="Types:Port" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TriPortIdListType">
    <xsd:sequence>
      <xsd:element name="comp" type="Types:TriPortIdType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Port">
    <xsd:sequence>
      <xsd:element name="id" type="Types:Id" />
      <xsd:element name="index" type="xsd:int" minOccurs="0" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TriComponentIdType">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="null" type="Templates:null"/>
        <xsd:element name="id" type="Types:Id" />
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TriComponentIdListType">
    <xsd:sequence>
      <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

```

```

<!-- Communication -->
<xsd:complexType name="TriMessageType">
  <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>

<xsd:complexType name="TriParameterType">
  <xsd:sequence>
    <xsd:element name="val" type="xsd:hexBinary" />
  </xsd:sequence>
  <xsd:attribute name="name" type="SimpleTypes:TString"/>
  <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>

<xsd:complexType name="TriParameterListType">
  <xsd:sequence>
    <xsd:element name="par" type="Types:TriParameterType" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TriExceptionType">
  <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>

<xsd:complexType name="TciValueList">
  <xsd:complexContent>
    <xsd:extension base="Values:RecordValue"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="TriSignatureIdType">
  <xsd:attribute name="val" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>

<xsd:complexType name="TriAddressType">
  <xsd:attribute name="val" type="SimpleTypes:TString"/>
</xsd:complexType>

<xsd:complexType name="TriAddressListType">
  <xsd:sequence>
    <xsd:element name="addr" type="Types:TriAddressType" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- Miscellaneous -->
<xsd:complexType name="TriTimerIdType">
  <xsd:sequence>
    <xsd:element name="id" type="Types:Id" />
  </xsd:sequence>
</xsd:complexType>

<!-- Basic definitions -->
<xsd:complexType name="QualifiedName">
  <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
  <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>

<!-- general TCI abstract data types -->
<xsd:complexType name="TciBehaviourIdType">
  <xsd:sequence>
    <xsd:element name="name" type="Types:QualifiedName" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TciTestCaseIdType">
  <xsd:sequence>
    <xsd:element name="name" type="Types:QualifiedName" />
  </xsd:sequence>
</xsd:complexType>

```

```

<xsd:complexType name="TciParameterType">
  <xsd:sequence>
    <xsd:element name="val" type="Values:Value" />
  </xsd:sequence>
  <xsd:attribute name="name" type="SimpleTypes:TString"/>
  <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>

<xsd:complexType name="TciParameterListType">
  <xsd:sequence>
    <xsd:element name="par" type="Types:TciParameterType" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- general identifier structure for test components, ports and timer -->
<xsd:complexType name="Id">
  <xsd:sequence>
    <xsd:element name="name" type="SimpleTypes:TString" />
    <xsd:element name="id" type="SimpleTypes:TString" minOccurs="0"/>
    <xsd:element name="type" type="SimpleTypes:TString" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

B.3 TCI-TL XML Schema for Values

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  xmlns:Values="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  xmlns:Templates="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
  xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  elementFormDefault="qualified">

  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
    schemaLocation="Templates_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
    schemaLocation="SimpleTypes_v3_3_1.xsd"/>

  <xsd:attributeGroup name="ValueAtts">
    <xsd:attribute name="name" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="type" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="module" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="annotation" type="SimpleTypes:TString" use="optional"/>
  </xsd:attributeGroup>

  <xsd:complexType name="Value" mixed="true">
    <xsd:choice>
      <xsd:element name="integer" type="Values:IntegerValue"/>
      <xsd:element name="float" type="Values:FloatValue"/>
      <xsd:element name="boolean" type="Values:BooleanValue"/>
      <xsd:element name="verdicttype" type="Values:VerdictValue"/>
      <xsd:element name="bitstring" type="Values:BitstringValue"/>
      <xsd:element name="hexstring" type="Values:HexstringValue"/>
      <xsd:element name="octetstring" type="Values:OctetstringValue"/>
      <xsd:element name="charstring" type="Values:CharstringValue"/>
      <xsd:element name="universal_charstring" type="Values:UniversalCharstringValue"/>
      <xsd:element name="record" type="Values:RecordValue"/>
      <xsd:element name="record_of" type="Values:RecordOfValue"/>
      <xsd:element name="array" type="Values:ArrayValue"/>
      <xsd:element name="set" type="Values:SetValue"/>
      <xsd:element name="set_of" type="Values:SetOfValue"/>
      <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
      <xsd:element name="union" type="Values:UnionValue"/>
      <xsd:element name="anytype" type="Values:AnytypeValue"/>
      <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
  </xsd:complexType>

```

```

<!-- general event elements -->
<xsd:complexType name="IntegerValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="FloatValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="BooleanValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="VerdictValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="BitstringValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="HexstringValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="OctetstringValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="CharstringValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="UniversalCharstringValue">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>

```

```

    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="RecordValue">
  <xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="integer" type="Values:IntegerValue"/>
      <xsd:element name="float" type="Values:FloatValue"/>
      <xsd:element name="boolean" type="Values:BooleanValue"/>
      <xsd:element name="verdicttype" type="Values:VerdictValue"/>
      <xsd:element name="bitstring" type="Values:BitstringValue"/>
      <xsd:element name="hexstring" type="Values:HexstringValue"/>
      <xsd:element name="octetstring" type="Values:OctetstringValue"/>
      <xsd:element name="charstring" type="Values:CharstringValue"/>
      <xsd:element name="universal_charstring"
        type="Values:UniversalCharstringValue"/>
      <xsd:element name="record" type="Values:RecordValue"/>
      <xsd:element name="record_of" type="Values:RecordOfValue"/>
      <xsd:element name="array" type="Values:ArrayValue"/>
      <xsd:element name="set" type="Values:SetValue"/>
      <xsd:element name="set_of" type="Values:SetOfValue"/>
      <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
      <xsd:element name="union" type="Values:UnionValue"/>
      <xsd:element name="anytype" type="Values:AnytypeValue"/>
      <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="RecordOfValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="charstring" type="Values:CharstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="array" type="Values:ArrayValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="set" type="Values:SetValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="set_of" type="Values:SetOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

```

<xsd:complexType name="ArrayValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="charstring" type="Values:CharstringValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="array" type="Values:ArrayValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="set" type="Values:SetValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="set_of" type="Values:SetOfValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="SetValue">
  <xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="integer" type="Values:IntegerValue"/>
      <xsd:element name="float" type="Values:FloatValue"/>
      <xsd:element name="boolean" type="Values:BooleanValue"/>
      <xsd:element name="verdicttype" type="Values:VerdictValue"/>
      <xsd:element name="bitstring" type="Values:BitstringValue"/>
      <xsd:element name="hexstring" type="Values:HexstringValue"/>
      <xsd:element name="octetstring" type="Values:OctetstringValue"/>
      <xsd:element name="charstring" type="Values:CharstringValue"/>
      <xsd:element name="universal_charstring"
        type="Values:UniversalCharstringValue"/>
      <xsd:element name="record" type="Values:RecordValue"/>
      <xsd:element name="record_of" type="Values:RecordOfValue"/>
      <xsd:element name="array" type="Values:ArrayValue"/>
      <xsd:element name="set" type="Values:SetValue"/>
      <xsd:element name="set_of" type="Values:SetOfValue"/>
      <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
      <xsd:element name="union" type="Values:UnionValue"/>
      <xsd:element name="anytype" type="Values:AnytypeValue"/>
      <xsd:element name="address" type="Values:AddressValue"/>
    </xsd:choice>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="SetOfValue">
  <xsd:choice>
    <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"

```

```

        maxOccurs="unbounded"/>
<xsd:element name="float" type="Values:FloatValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="verdicttype" type="Values:VerdictValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="bitstring" type="Values:BitstringValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="hexstring" type="Values:HexstringValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="octetstring" type="Values:OctetstringValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="charstring" type="Values:CharstringValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="universal_charstring"
type="Values:UniversalCharstringValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record" type="Values:RecordValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record_of" type="Values:RecordOfValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="array" type="Values:ArrayValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="set" type="Values:SetValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="set_of" type="Values:SetOfValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="enumerated" type="Values:EnumeratedValue"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="union" type="Values:UnionValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="address" type="Values:AddressValue" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="EnumeratedValue">
<xsd:choice>
<xsd:element name="value" type="SimpleTypes:TString"/>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="UnionValue">
<xsd:choice>
<xsd:element name="integer" type="Values:IntegerValue"/>
<xsd:element name="float" type="Values:FloatValue"/>
<xsd:element name="boolean" type="Values:BooleanValue"/>
<xsd:element name="verdicttype" type="Values:VerdictValue"/>
<xsd:element name="bitstring" type="Values:BitstringValue"/>
<xsd:element name="hexstring" type="Values:HexstringValue"/>
<xsd:element name="octetstring" type="Values:OctetstringValue"/>
<xsd:element name="charstring" type="Values:CharstringValue"/>
<xsd:element name="universal_charstring"
type="Values:UniversalCharstringValue"/>
<xsd:element name="record" type="Values:RecordValue"/>
<xsd:element name="record_of" type="Values:RecordOfValue"/>
<xsd:element name="array" type="Values:ArrayValue"/>
<xsd:element name="set" type="Values:SetValue"/>
<xsd:element name="set_of" type="Values:SetOfValue"/>
<xsd:element name="enumerated" type="Values:EnumeratedValue"/>
<xsd:element name="union" type="Values:UnionValue"/>
<xsd:element name="anytype" type="Values:AnytypeValue"/>
<xsd:element name="address" type="Values:AddressValue"/>
<xsd:element name="null" type="Templates:null"/>
<xsd:element name="omit" type="Templates:omit"/>
</xsd:choice>
<xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

```

```

<xsd:complexType name="AnytypeValue">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:OctetstringValue"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
    <xsd:element name="set" type="Values:SetValue"/>
    <xsd:element name="set_of" type="Values:SetOfValue"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
    <xsd:element name="union" type="Values:UnionValue"/>
    <xsd:element name="address" type="Values:AddressValue"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="AddressValue">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="integer" type="Values:IntegerValue"/>
    <xsd:element name="float" type="Values:FloatValue"/>
    <xsd:element name="boolean" type="Values:BooleanValue"/>
    <xsd:element name="verdicttype" type="Values:VerdictValue"/>
    <xsd:element name="bitstring" type="Values:BitstringValue"/>
    <xsd:element name="hexstring" type="Values:HexstringValue"/>
    <xsd:element name="octetstring" type="Values:OctetstringValue"/>
    <xsd:element name="charstring" type="Values:OctetstringValue"/>
    <xsd:element name="universal_charstring"
      type="Values:UniversalCharstringValue"/>
    <xsd:element name="record" type="Values:RecordValue"/>
    <xsd:element name="record_of" type="Values:RecordOfValue"/>
    <xsd:element name="array" type="Values:ArrayValue"/>
    <xsd:element name="set" type="Values:SetValue"/>
    <xsd:element name="set_of" type="Values:SetOfValue"/>
    <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
    <xsd:element name="union" type="Values:UnionValue"/>
    <xsd:element name="anytype" type="Values:AnytypeValue"/>
    <xsd:element name="null" type="Templates:null"/>
    <xsd:element name="omit" type="Templates:omit"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
</xsd:schema>

```

B.4 TCI-TL XML Schema for Templates

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
  xmlns:Templates="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
  xmlns:Values="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  xmlns:Types="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
  xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  elementFormDefault="qualified">

  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
    schemaLocation="Values_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
    schemaLocation="Types_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
    schemaLocation="SimpleTypes_v3_3_1.xsd"/>

  <xsd:complexType name="TciValueTemplate">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Values:Value">
        <xsd:choice>

```



```

<xsd:element name="integer" type="Templates:IntegerTemplate"/>
<xsd:element name="float" type="Templates:FloatTemplate"/>
<xsd:element name="boolean" type="Templates:BooleanTemplate"/>
<xsd:element name="verdicttype" type="Templates:VerdictTemplate"/>
<xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
<xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
<xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
<xsd:element name="charstring" type="Templates:CharstringTemplate"/>
<xsd:element name="universal_charstring"
  type="Templates:UniversalCharstringTemplate"/>
<xsd:element name="record" type="Templates:RecordTemplate"/>
<xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
<xsd:element name="array" type="Values:ArrayValue"/>
<xsd:element name="set" type="Templates:SetTemplate"/>
<xsd:element name="set_of" type="Templates:SetOfTemplate"/>
<xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
<xsd:element name="union" type="Templates:UnionTemplate"/>
<xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
<xsd:element name="address" type="Templates:AddressTemplate"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
</xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="omit">
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="any">
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="anyoromit">
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="TciNonValueTemplate">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="any" type="Templates:any"/>
      <xsd:element name="all" type="Templates:all"/>
      <xsd:element name="templateDef" type="SimpleTypes:TString"/>
      <xsd:element name="null" type="Templates:null"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="all">
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="null">
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="TciValueDifference">
  <xsd:sequence>
    <xsd:element name="val" type="SimpleTypes:xpath"/>
    <xsd:element name="tmpl" type="SimpleTypes:xpath"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
  <xsd:attribute name="desc" type="SimpleTypes:TString" use="optional"/>
</xsd:complexType>

<xsd:complexType name="TciValueDifferenceList">
  <xsd:sequence>
    <xsd:element name="diff" type="Templates:TciValueDifference"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="IntegerTemplate">
  <xsd:choice>

```



```

    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="UniversalCharstringTemplate">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
    <xsd:element name="omit" type="Templates:omit"/>
    <xsd:element name="any" type="Templates:any"/>
    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
    <xsd:element name="null" type="Templates:null"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="VerdictTemplate">
  <xsd:choice>
    <xsd:element name="value" type="SimpleTypes:TString"/>
    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
    <xsd:element name="omit" type="Templates:omit"/>
    <xsd:element name="any" type="Templates:any"/>
    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
    <xsd:element name="null" type="Templates:null"/>
  </xsd:choice>
  <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="RecordTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:RecordValue">
      <xsd:choice>
        <xsd:choice minOccurs="0">
          <xsd:element name="integer" type="Templates:IntegerTemplate"/>
          <xsd:element name="float" type="Templates:FloatTemplate"/>
          <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
          <xsd:element name="verdicttype" type="Templates:VerdictTemplate"/>
          <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
          <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
          <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
          <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
          <xsd:element name="universal_charstring"
            type="Templates:UniversalCharstringTemplate"/>
          <xsd:element name="record" type="Templates:RecordTemplate"/>
          <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
          <xsd:element name="array" type="Values:ArrayValue"/>
          <xsd:element name="set" type="Templates:SetTemplate"/>
          <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
          <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
          <xsd:element name="union" type="Templates:UnionTemplate"/>
          <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
          <xsd:element name="address" type="Templates:AddressTemplate"/>
        </xsd:choice>
        <xsd:element name="omit" type="Templates:omit"/>
        <xsd:element name="any" type="Templates:any"/>
        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
        <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RecordOfTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:RecordOfValue">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="verdicttype" type="Templates:VerdictTemplate" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="bitstring" type="Templates:BitstringTemplate"
          minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="hexstring" type="Templates:HexstringTemplate"

```

```

        minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="octetstring" type="Templates:OctetstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="charstring" type="Templates:CharstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record_of" type="Templates:RecordOfTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="array" type="Templates:ArrayTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="set_of" type="Templates:SetOfTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
<xsd:element name="null" type="Templates:null"/>
</xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:SetValue">
      <xsd:choice>
        <xsd:choice minOccurs="0">
          <xsd:element name="integer" type="Templates:IntegerTemplate"/>
          <xsd:element name="float" type="Templates:FloatTemplate"/>
          <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
          <xsd:element name="verdictctype" type="Templates:VerdictTemplate"/>
          <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
          <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
          <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
          <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
          <xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate"/>
          <xsd:element name="record" type="Templates:RecordTemplate"/>
          <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
          <xsd:element name="array" type="Templates:ArrayTemplate"/>
          <xsd:element name="set" type="Templates:SetTemplate"/>
          <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
          <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
          <xsd:element name="union" type="Templates:UnionTemplate"/>
          <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
          <xsd:element name="address" type="Templates:AddressTemplate"/>
        </xsd:choice>
        <xsd:element name="omit" type="Templates:omit"/>
        <xsd:element name="any" type="Templates:any"/>
        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
        <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetOfTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:SetOfValue">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"

```

```

        maxOccurs="unbounded"/>
<xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="verdicttype" type="Templates:VerdictTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="bitstring" type="Templates:BitstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="hexstring" type="Templates:HexstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="octetstring" type="Templates:OctetstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="charstring" type="Templates:CharstringTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="record_of" type="Templates:RecordOfTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="array" type="Templates:ArrayTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="set_of" type="Templates:SetOfTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
<xsd:element name="null" type="Templates:null"/>
</xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="EnumeratedTemplate">
<xsd:complexContent>
<xsd:extension base="Values:EnumeratedValue">
<xsd:choice minOccurs="0">
<xsd:element name="value" type="SimpleTypes:TString"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="null" type="Templates:null"/>
</xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UnionTemplate">
<xsd:complexContent>
<xsd:extension base="Values:UnionValue">
<xsd:choice minOccurs="0">
<xsd:element name="integer" type="Templates:IntegerTemplate"/>
<xsd:element name="float" type="Templates:FloatTemplate"/>
<xsd:element name="boolean" type="Templates:BooleanTemplate"/>
<xsd:element name="verdicttype" type="Templates:VerdictTemplate"/>
<xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
<xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
<xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
<xsd:element name="charstring" type="Templates:CharstringTemplate"/>
<xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate"/>
<xsd:element name="record" type="Templates:RecordTemplate"/>
<xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
<xsd:element name="array" type="Templates:ArrayTemplate"/>
<xsd:element name="set" type="Templates:SetTemplate"/>
<xsd:element name="set_of" type="Templates:SetOfTemplate"/>

```

```

<xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
<xsd:element name="union" type="Templates:UnionTemplate"/>
<xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
<xsd:element name="address" type="Templates:AddressTemplate"/>
<xsd:element name="omit" type="Templates:omit"/>
<xsd:element name="any" type="Templates:any"/>
<xsd:element name="anyoromit" type="Templates:anyoromit"/>
<xsd:element name="templateDef" type="SimpleTypes:TString"/>
<xsd:element name="null" type="Templates:null"/>
  </xsd:choice>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="AnytypeTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:AnytypeValue">
      <xsd:choice minOccurs="0">
        <xsd:element name="integer" type="Templates:IntegerTemplate"/>
        <xsd:element name="float" type="Templates:FloatTemplate"/>
        <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
        <xsd:element name="verdicttype" type="Templates:VerdictTemplate"/>
        <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
        <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
        <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
        <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
        <xsd:element name="universal_charstring"
          type="Templates:UniversalCharstringTemplate"/>
        <xsd:element name="record" type="Templates:RecordTemplate"/>
        <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
        <xsd:element name="array" type="Templates:ArrayTemplate"/>
        <xsd:element name="set" type="Templates:SetTemplate"/>
        <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
        <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
        <xsd:element name="union" type="Templates:UnionTemplate"/>
        <xsd:element name="address" type="Templates:AddressTemplate"/>
        <xsd:element name="omit" type="Templates:omit"/>
        <xsd:element name="any" type="Templates:any"/>
        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
        <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="AddressTemplate">
  <xsd:complexContent>
    <xsd:extension base="Values:AnytypeValue">
      <xsd:choice minOccurs="0">
        <xsd:element name="integer" type="Templates:IntegerTemplate"/>
        <xsd:element name="float" type="Templates:FloatTemplate"/>
        <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
        <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
        <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
        <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
        <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
        <xsd:element name="universal_charstring"
          type="Templates:UniversalCharstringTemplate"/>
        <xsd:element name="record" type="Templates:RecordTemplate"/>
        <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
        <xsd:element name="array" type="Templates:ArrayTemplate"/>
        <xsd:element name="set" type="Templates:SetTemplate"/>
        <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
        <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
        <xsd:element name="union" type="Templates:UnionTemplate"/>
        <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
        <xsd:element name="omit" type="Templates:omit"/>
        <xsd:element name="any" type="Templates:any"/>
        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
        <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        <xsd:element name="null" type="Templates:null"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>

```

B.5 TCI-TL XML Schema for Events

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/Events_v3_3_1.xsd"
  xmlns:Events="http://uri.etsi.org/ttcn-3/tci/Events_v3_3_1.xsd"
  xmlns:Types="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
  xmlns:Templates="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
  xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
  xmlns:Values="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  elementFormDefault="qualified">

  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/SimpleTypes_v3_3_1.xsd"
    schemaLocation="SimpleTypes_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
    schemaLocation="Types_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
    schemaLocation="Values_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Templates_v3_3_1.xsd"
    schemaLocation="Templates_v3_3_1.xsd"/>

  <!-- common definition for all events -->
  <xsd:complexType name="Event" mixed="true">
    <xsd:sequence>
      <xsd:element name="am" type="SimpleTypes:TString"/>
    </xsd:sequence>
    <xsd:attribute name="ts" type="xsd:long" use="required"/>
    <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

    <!-- general identifier structure for test components, ports and timer -->
    <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
    <xsd:attribute name="id" type="SimpleTypes:TString" use="required"/>
    <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
  </xsd:complexType>

  <!-- this event is extended by all port configuration events -->
  <xsd:complexType name="PortConfiguration">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="port1" type="Types:TriPortIdType" />
          <xsd:element name="port2" type="Types:TriPortIdType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- this event is extended by all port status events -->
  <xsd:complexType name="PortStatus">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="port" type="Types:TriPortIdType"/>
          <xsd:element name="stat" type="SimpleTypes:PortStatusType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- testcases -->
  <xsd:complexType name="tliTcExecute">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliTcStart">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>

```

```

        <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcStop">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcStarted">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTcTerminated">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="verdict" type="Values:VerdictValue"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- control -->
<xsd:complexType name="tliCtrlStart">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCtrlStop">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCtrlTerminated">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<!-- asynchronous communication -->
<xsd:complexType name="tliMSend_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="msgValue" type="Values:Value"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```



```

    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_m_BC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:choice>
            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
            <xsd:sequence>
              <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
              <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
            </xsd:sequence>
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_m_MC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
          <xsd:choice>
            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
            <xsd:sequence>
              <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
              <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
              <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
            </xsd:sequence>
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c_BC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="msgValue" type="Values:Value"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMSend_c_MC">
    <xsd:complexContent mixed="true">

```

```

    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMDetected_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Types:TriMessageType"/>
        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMDetected_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="msgValue" type="Values:Value"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMMismatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:element name="msgTpl" type="Templates:TciValueTemplate"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMMismatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="msgValue" type="Values:Value"/>
        <xsd:element name="msgTpl" type="Templates:TciValueTemplate"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMReceive_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="msgTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>

```

```

        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliMReceive_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- synchronous communication -->
<xsd:complexType name="tliPrCall_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_m_BC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_m_MC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
            <xsd:sequence>
                <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
                <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:choice>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_c_BC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCall_c_MC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCallDetected_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="triPars" type="Types:TriParameterListType" minOccurs="0"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:complexType name="tliPrGetCallDetected_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCallMismatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCallMismatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCall_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetCall_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:complexType name="tliPrReply_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
            minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
              <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
              <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
              <xsd:element name="transmission-failure"
                type="SimpleTypes:TriStatusType" minOccurs="0"/>
            </xsd:sequence>
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrReply_m_BC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
          <xsd:choice>
            <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
              minOccurs="0"/>
            <xsd:sequence>
              <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                <xsd:element name="transmission-failure"
                  type="SimpleTypes:TriStatusType" minOccurs="0"/>
              </xsd:sequence>
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_MC">
      <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
          <xsd:sequence>
            <xsd:element name="at" type="Types:TriPortIdType"/>
            <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
            <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
            <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
            <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
            <xsd:choice>
              <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
                minOccurs="0"/>
              <xsd:sequence>
                <xsd:element name="triPars" type="Types:TriParameterListType"
minOccurs="0"/>
                  <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                  <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
                    <xsd:element name="transmission-failure"
                      type="SimpleTypes:TriStatusType" minOccurs="0"/>
                </xsd:sequence>
              </xsd:choice>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

```

```

    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrReply_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrReply_c_BC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrReply_c_MC">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
          <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetReplyDetected_m">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="triPars" type="Types:TriParameterListType" minOccurs="0"/>
          <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
          <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliPrGetReplyDetected_c">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="at" type="Types:TriPortIdType"/>
          <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
          <xsd:element name="signature" type="Types:TriSignatureIdType"/>
          <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
          <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

```

```

</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_m">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="parsTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="replValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="replTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m">
  <xsd:complexContent mixed="true">

```



```

    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
            <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m_BC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m_MC">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>
        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addrValues" type="Types:TciValueListType" minOccurs="0"/>
        <xsd:choice>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:sequence>
            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
            <xsd:element name="addresses" type="Types:TriAddressListType"
minOccurs="0"/>
            <xsd:element name="transmission-failure"
type="SimpleTypes:TciStatusType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="at" type="Types:TriPortIdType"/>
        <xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>

```

```

        <xsd:element name="signature" type="Types:TriSignatureIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c_BC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c_MC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

        <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="addrValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="excValue" type="Values:Value" minOccurs="0"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate" minOccurs="0"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeoutDetected">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeout">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="at" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

<!-- components -->
<xsd:complexType name="tliCCreate">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="name" type="SimpleTypes:TString"/>
        <xsd:element name="alive" type="SimpleTypes:TBoolean"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStart">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="name" type="Types:TciBehaviourIdType"/>
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCRunning">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCAlive">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="status" type="SimpleTypes:ComponentStatusType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStop">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKill">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDoneMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>

```

```

</xsd:complexType>

<xsd:complexType name="tliCKilledMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType"/>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDone">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKilled">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="compTpl" type="Templates:TciNonValueTemplate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCTerminated">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="verdict" type="Values:VerdictValue" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- ports -->
<xsd:complexType name="tliPConnect">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortConfiguration"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPDisconnect">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortConfiguration"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPMap">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortConfiguration"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPUnmap">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortConfiguration"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPClear">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortStatus"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStart">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortStatus"/>
  </xsd:complexContent>

```

```

</xsd:complexType>

<xsd:complexType name="tliPStop">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortStatus"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPHalt">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:PortStatus"/>
  </xsd:complexContent>
</xsd:complexType>

<!-- codec -->
<xsd:complexType name="tliEncode">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="val" type="Values:Value"/>
        <xsd:choice>
          <xsd:element name="msg" type="Types:TriMessageType"/>
          <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
        </xsd:choice>
        <xsd:element name="codec" type="SimpleTypes:TString"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliDecode" mixed="true">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="msg" type="Types:TriMessageType"/>
        <xsd:choice>
          <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
          <xsd:element name="val" type="Values:Value"/>
        </xsd:choice>
        <xsd:element name="codec" type="SimpleTypes:TString"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- timers -->
<xsd:complexType name="tliTimeoutDetected">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTimeoutMismatch">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType" />
        <xsd:element name="timerTpl" type="Templates:TciNonValueTemplate" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTimeout">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType" />
        <xsd:element name="timerTpl" type="Templates:TciNonValueTemplate" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTStart">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTStop">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType"/>
        <xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTRead">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType"/>
        <xsd:element name="elapsed" type="SimpleTypes:TriTimerDurationType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliTRunning">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="timer" type="Types:TriTimerIdType"/>
      </xsd:sequence>
      <xsd:attribute name="status" type="SimpleTypes:TimerStatusType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- scope -->
<xsd:complexType name="tliSEnter">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="kind" type="SimpleTypes:TString"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliSLeave">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">
      <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" />
        <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
        <xsd:element name="returnValue" type="Values:Value" minOccurs="0"/>
        <xsd:element name="kind" type="SimpleTypes:TString"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- variables and module parameter -->
<xsd:complexType name="tliVar">
  <xsd:complexContent mixed="true">
    <xsd:extension base="Events:Event">

```

```

        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName" />
            <xsd:element name="val" type="Values:Value" minOccurs="0"/>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexType>

<xsd:complexType name="tliModulePar">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" />
                <xsd:element name="val" type="Values:Value" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- verdicts -->
<xsd:complexType name="tliGetVerdict">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="verdict" type="Values:VerdictValue"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliSetVerdict">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="verdict" type="Values:VerdictValue"/>
                <xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- log -->
<xsd:complexType name="tliLog">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="log" type="SimpleTypes:TString"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- alt -->
<xsd:complexType name="tliAEnter">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliALeave">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliADefaults">
    <xsd:complexContent>
        <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliAActivate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" />
                <xsd:element name="tciPars" type="Types:TciParameterListType" minOccurs="0"/>
                <xsd:element name="ref" type="Values:Value"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```



```

        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliADeactivate">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="ref" type="Values:Value"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliANomatch">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliARepeat">
    <xsd:complexContent>
      <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliAwait">
    <xsd:complexContent>
      <xsd:extension base="Events:Event"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliAction">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="action" type="SimpleTypes:TString"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMatch">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="expr" type="Values:Value"/>
          <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliMatchMismatch">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="expr" type="Values:Value"/>
          <xsd:element name="tmpl" type="Templates:TciValueTemplate"/>
          <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="tliInfo">
    <xsd:complexContent mixed="true">
      <xsd:extension base="Events:Event">
        <xsd:sequence>
          <xsd:element name="level" type="SimpleTypes:TInteger"/>
          <xsd:element name="info" type="SimpleTypes:TString"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>

```

B.6 TCI-TL XML Schema for a Log

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://uri.etsi.org/ttcn-3/tci/TLI_v3_3_1.xsd"
  xmlns:TLI="http://uri.etsi.org/ttcn-3/tci/TLI_v3_3_1.xsd"
  xmlns:Types="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
  xmlns:Values="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
  xmlns:Events="http://uri.etsi.org/ttcn-3/tci/Events_v3_3_1.xsd"
  elementFormDefault="qualified">

  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Types_v3_3_1.xsd"
    schemaLocation="Types_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Values_v3_3_1.xsd"
    schemaLocation="Values_v3_3_1.xsd"/>
  <xsd:import namespace="http://uri.etsi.org/ttcn-3/tci/Events_v3_3_1.xsd"
    schemaLocation="Events_v3_3_1.xsd"/>

  <xsd:element name="logfile" type="TLI:LogModule"/>
  <xsd:complexType name="LogModule">
    <xsd:sequence>
      <xsd:element name="header" type="TLI:Header"/>
      <xsd:element name="body" type="TLI:Body"/>
      <xsd:element name="trailer" type="TLI:Trailer" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Header">
    <xsd:sequence>
      <!-- logging version -->
      <xsd:element name="version" type="xsd:string"/>
      <!-- begin of the log -->
      <xsd:element name="ts" type="xsd:long"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Trailer">
    <xsd:choice>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="Body">
    <xsd:choice maxOccurs="unbounded">

      <!-- test cases operations -->
      <xsd:element name="tliTcExecute" type="Events:tliTcExecute"/>
      <xsd:element name="tliTcStart" type="Events:tliTcStart"/>
      <xsd:element name="tliTcStop" type="Events:tliTcStop"/>
      <xsd:element name="tliTcStarted" type="Events:tliTcStarted"/>
      <xsd:element name="tliTcTerminated" type="Events:tliTcTerminated"/>

      <!-- control operations -->
      <xsd:element name="tliCtrlStart" type="Events:tliCtrlStart"/>
      <xsd:element name="tliCtrlStop" type="Events:tliCtrlStop"/>
      <xsd:element name="tliCtrlTerminated" type="Events:tliCtrlTerminated"/>

      <!-- asynchronous communication -->
      <xsd:element name="tliMSend_m" type="Events:tliMSend_m"/>
      <xsd:element name="tliMSend_c" type="Events:tliMSend_c"/>
      <xsd:element name="tliMSend_m_BC" type="Events:tliMSend_m_BC"/>
      <xsd:element name="tliMSend_c_BC" type="Events:tliMSend_c_BC"/>
      <xsd:element name="tliMSend_m_MC" type="Events:tliMSend_m_MC"/>
      <xsd:element name="tliMSend_c_MC" type="Events:tliMSend_c_MC"/>
      <xsd:element name="tliMDetected_m" type="Events:tliMDetected_m"/>
      <xsd:element name="tliMDetected_c" type="Events:tliMDetected_c"/>
      <xsd:element name="tliMMismatch_m" type="Events:tliMMismatch_m"/>
      <xsd:element name="tliMMismatch_c" type="Events:tliMMismatch_c"/>
      <xsd:element name="tliMReceive_m" type="Events:tliMReceive_m"/>
      <xsd:element name="tliMReceive_c" type="Events:tliMReceive_c"/>

      <!-- synchronous communication -->
      <xsd:element name="tliPrCall_m" type="Events:tliPrCall_m"/>
      <xsd:element name="tliPrCall_c" type="Events:tliPrCall_c"/>
      <xsd:element name="tliPrCall_m_BC" type="Events:tliPrCall_m_BC"/>
      <xsd:element name="tliPrCall_c_BC" type="Events:tliPrCall_c_BC"/>
      <xsd:element name="tliPrCall_m_MC" type="Events:tliPrCall_m_MC"/>
      <xsd:element name="tliPrCall_c_MC" type="Events:tliPrCall_c_MC"/>
    </xsd:choice>
  </xsd:complexType>

```

```

<xsd:element name="tliPrGetCallDetected_m" type="Events:tliPrGetCallDetected_m"/>
<xsd:element name="tliPrGetCallDetected_c" type="Events:tliPrGetCallDetected_c"/>
<xsd:element name="tliPrGetCallMismatch_m" type="Events:tliPrGetCallMismatch_m"/>
<xsd:element name="tliPrGetCallMismatch_c" type="Events:tliPrGetCallMismatch_c"/>
<xsd:element name="tliPrGetCall_m" type="Events:tliPrGetCall_m"/>
<xsd:element name="tliPrGetCall_c" type="Events:tliPrGetCall_c"/>

<xsd:element name="tliPrReply_m" type="Events:tliPrReply_m"/>
<xsd:element name="tliPrReply_c" type="Events:tliPrReply_c"/>
<xsd:element name="tliPrReply_m_BC" type="Events:tliPrReply_m_BC"/>
<xsd:element name="tliPrReply_c_BC" type="Events:tliPrReply_c_BC"/>
<xsd:element name="tliPrReply_m_MC" type="Events:tliPrReply_m_MC"/>
<xsd:element name="tliPrReply_c_MC" type="Events:tliPrReply_c_MC"/>

<xsd:element name="tliPrGetReplyDetected_m" type="Events:tliPrGetReplyDetected_m"/>
<xsd:element name="tliPrGetReplyDetected_c" type="Events:tliPrGetReplyDetected_c"/>
<xsd:element name="tliPrGetReplyMismatch_m" type="Events:tliPrGetReplyMismatch_m"/>
<xsd:element name="tliPrGetReplyMismatch_c" type="Events:tliPrGetReplyMismatch_c"/>
<xsd:element name="tliPrGetReply_m" type="Events:tliPrGetReply_m"/>
<xsd:element name="tliPrGetReply_c" type="Events:tliPrGetReply_c"/>

<xsd:element name="tliPrRaise_m" type="Events:tliPrRaise_m"/>
<xsd:element name="tliPrRaise_c" type="Events:tliPrRaise_c"/>
<xsd:element name="tliPrRaise_m_BC" type="Events:tliPrRaise_m_BC"/>
<xsd:element name="tliPrRaise_c_BC" type="Events:tliPrRaise_c_BC"/>
<xsd:element name="tliPrRaise_m_MC" type="Events:tliPrRaise_m_MC"/>
<xsd:element name="tliPrRaise_c_MC" type="Events:tliPrRaise_c_MC"/>

<xsd:element name="tliPrCatchDetected_m" type="Events:tliPrCatchDetected_m"/>
<xsd:element name="tliPrCatchDetected_c" type="Events:tliPrCatchDetected_c"/>
<xsd:element name="tliPrCatchMismatch_m" type="Events:tliPrCatchMismatch_m"/>
<xsd:element name="tliPrCatchMismatch_c" type="Events:tliPrCatchMismatch_c"/>
<xsd:element name="tliPrCatch_m" type="Events:tliPrCatch_m"/>
<xsd:element name="tliPrCatch_c" type="Events:tliPrCatch_c"/>

<xsd:element name="tliPrCatchTimeoutDetected"
                type="Events:tliPrCatchTimeoutDetected" />
<xsd:element name="tliPrCatchTimeout" type="Events:tliPrCatchTimeout"/>

<!-- components -->
<xsd:element name="tliCCreate" type="Events:tliCCreate"/>
<xsd:element name="tliCStart" type="Events:tliCStart"/>
<xsd:element name="tliCRunning" type="Events:tliCRunning"/>
<xsd:element name="tliCAlive" type="Events:tliCAlive"/>
<xsd:element name="tliCStop" type="Events:tliCStop"/>
<xsd:element name="tliCKill" type="Events:tliCKill"/>
<xsd:element name="tliCDoneMismatch" type="Events:tliCDoneMismatch"/>
<xsd:element name="tliCDone" type="Events:tliCDone"/>
<xsd:element name="tliCKilledMismatch" type="Events:tliCKilledMismatch"/>
<xsd:element name="tliCKilled" type="Events:tliCKilled"/>
<xsd:element name="tliCTerminated" type="Events:tliCTerminated"/>

<!-- ports -->
<xsd:element name="tliPConnect" type="Events:tliPConnect"/>
<xsd:element name="tliPDisconnect" type="Events:tliPDisconnect"/>
<xsd:element name="tliPMap" type="Events:tliPMap"/>
<xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>
<xsd:element name="tliPClear" type="Events:tliPClear"/>
<xsd:element name="tliPStart" type="Events:tliPStart"/>
<xsd:element name="tliPStop" type="Events:tliPStop"/>
<xsd:element name="tliPHalt" type="Events:tliPHalt"/>

<!-- codec -->
<xsd:element name="tliDecode" type="Events:tliDecode"/>
<xsd:element name="tliEncode" type="Events:tliEncode"/>

<!-- timers -->
<xsd:element name="tliTTimeoutDetected" type="Events:tliTTimeoutDetected"/>
<xsd:element name="tliTTimeoutMismatch" type="Events:tliTTimeoutMismatch"/>
<xsd:element name="tliTTimeout" type="Events:tliTTimeout"/>
<xsd:element name="tliTStart" type="Events:tliTStart"/>
<xsd:element name="tliTStop" type="Events:tliTStop"/>
<xsd:element name="tliTRead" type="Events:tliTRead"/>
<xsd:element name="tliTRunning" type="Events:tliTRunning"/>

<!-- scopes -->
<xsd:element name="tliSEnter" type="Events:tliSEnter"/>
<xsd:element name="tliSLeave" type="Events:tliSLeave"/>

```

```
<!-- statements -->
<xsd:element name="tliVar" type="Events:tliVar"/>
<xsd:element name="tliModulePar" type="Events:tliModulePar"/>
<xsd:element name="tliGetVerdict" type="Events:tliGetVerdict"/>
<xsd:element name="tliSetVerdict" type="Events:tliSetVerdict"/>
<xsd:element name="tliLog" type="Events:tliLog"/>

<!-- alt -->
<xsd:element name="tliAEnter" type="Events:tliAEnter"/>
<xsd:element name="tliALeave" type="Events:tliALeave"/>
<xsd:element name="tliADefaults" type="Events:tliADefaults"/>
<xsd:element name="tliAActivate" type="Events:tliAActivate"/>
<xsd:element name="tliADeactivate" type="Events:tliADeactivate"/>
<xsd:element name="tliANomatch" type="Events:tliANomatch"/>
<xsd:element name="tliARepeat" type="Events:tliARepeat"/>
<xsd:element name="tliAWait" type="Events:tliAWait"/>

<!-- action -->
<xsd:element name="tliAction" type="Events:tliAction"/>

<!-- match -->
<xsd:element name="tliMatch" type="Events:tliMatch"/>
<xsd:element name="tliMatchMismatch" type="Events:tliMatchMismatch"/>

<!-- info -->
<xsd:element name="tliInfo" type="Events:tliInfo"/>
</xsd:choice>
</xsd:complexType>
</xsd:schema>
```

Annex C (informative): Bibliography

- INTOOL CGI/NPL038 (V2.2): "Generic Compiler/Interpreter interface; GCI Interface Specification" Infrastructural Tools, December 1996.
- ISO/IEC 9646-3 (1998): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 3: The Tree and Tabular combined Notation (TTCN)".
- CORBA 3.0: "The Common Object Request Broker - Architecture and Specification", OMG Formal Document.

History

Document history		
V1.1.1	July 2003	Publication
V3.1.1	June 2005	Publication
V3.2.1	February 2007	Publication
V3.3.1	April 2008	Publication
V3.4.1	September 2008	Publication
V4.1.1	March 2009	Membership Approval Procedure MV 20090522: 2009-03-24 to 2009-05-22