ETSI ES 201 873-6 V4.11.1 (2019-04)

Methods for Testing and Specification (MTS);

The Testing and Test Control Notation version 3;

Part 6: TTCN‑3 Control Interface (TCI)

**ETSI Standard**

Reference

RES/MTS-201873-6ed4.11.1

Keywords

control, interface, methodology, TCI, testing, TTCN-3

***ETSI***

650 Route des Lucioles

F-06921 Sophia Antipolis Cedex - FRANCE

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

Siret N° 348 623 562 00017 - NAF 742 C

Association à but non lucratif enregistrée à la

Sous-Préfecture de Grasse (06) N° 7803/88

***Important notice***

The present document can be downloaded from:  
<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at <https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:  
<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

***Copyright Notification***

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

© ETSI 2019.

All rights reserved.

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

#### 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 TciParameterList \*tciPars, 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, const TString &reason)=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, const TString &reason)=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 enqueuing 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 enqueuing 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 \*msgTmpl, const TciValueDifferenceList \*diffs, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*msgTmpl, const TciValueDifferenceList \*diffs, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

// Called by TE to log the receiving of a message

virtual void tliMReceive\_m (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const TriPortId \*at, const TciValue \*msgValue, const TciValueTemplate \*msgTmpl, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*msgTmpl, const TriComponentId \*fromComp, const TciNonValueTemplate \*fromTmpl)=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 \*parsTmpl, const TciValueDifferenceList \*diffs, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*parsTmpl, const TciValueDifferenceList \*diffs, const TriComponentId \*from, const TciValueTemplate \*fromTmpl)=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 \*parsTmpl, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*parsTmpl, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=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 \*parsTmpl, const TciValue \*replValue, const TciValueTemplate \*replyTmpl, const TciValueDifferenceList \*diffs, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*parsTmpl, const TciValue \*replValue, const TciValueTemplate \*replyTmpl, const TciValueDifferenceList \*diffs, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=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 \*parsTmpl, const TciValue \*replValue, const TciValueTemplate \*replyTmpl, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*parsTmpl, const TciValue \*replValue, const TciValueTemplate \*replyTmpl, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=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 \*excTmpl, const TciValueDifferenceList \*diffs, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*excTmpl, const TciValueDifferenceList \*diffs, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=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 \*excTmpl, const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=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 \*excTmpl, const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=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 \*compTmpl)=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 \*compTmpl, const VerdictValue \* verdict)=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 \*compTmpl)=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 \*compTmpl, const VerdictValue \* verdict)=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, const TString &reason)=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 map operation including param

virtual void tliPMapParam (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const TriPortId \*port1, const TriPortId \*port2, const TciParameterList \*tciPars, const TriStatus \*encoderFailure,   
const TriParameterList \*triPars)=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 unmap operation including param

virtual void tliPUnmapParam (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const TriPortId \*port1, const TriPortId \*port2, const TciParameterList \*tciPars, const TriStatus \*encoderFailure,   
const TriParameterList \*triPars)=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 \*timerTmpl)=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 \*timerTmpl)=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 tliVar (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;

//Called by TE to log the checking of a message

virtual void tliMChecked\_m (const Tstring &am, const timeval ts, const Tstring &src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TciValue \*msgValue, const TciValueTemplate \*msgTmpl,

const TciValue \*address, const TciValueTemplate \*addressTmpl)=0;

//Called by CH to log the checking of a message

virtual void tliMChecked\_c (const Tstring &am, const timeval ts, const Tstring src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TciValue \*msgValue, const TciValueTemplate \*msgTmpl,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log checking of the getcall operation

virtual void tliPrGetCallChecked\_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 \*parsTmpl,

const TciValue \*address, const TciValueTemplate \*addressTmpl)=0;

//Called by TE to log checking of the getcall operation

virtual void tliPrGetCallChecked\_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 \*parsTmpl,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log checking of the getreply operation

virtual void tliPrGetReplyChecked\_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 \*parsTmpl,

const TciValue \*replValue, const TciValueTemplate \*replyTmpl,

const TciValue \*address, const TciValueTemplate \*addressTmpl)=0;

//Called by CH to log checking of the getreply operation

virtual void tliPrGetReplyChecked\_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 \*parsTmpl,

const TciValue \*replValue, const TciValueTemplate \*replyTmpl,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log checking of the catch operation

virtual void tliPrCatchChecked\_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 \*excTmpl,

const TciValue \*address, const TciValueTemplate \*addressTmpl)=0;

//Called by TE to log checking of the catch operation

virtual void tliPrCatchChecked\_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 \*excTmpl,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log the check any operation

virtual void tliCheckedAny\_m (const Tstring &am, const timeval ts, const Tstring &src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TciValue \*address, const TciValueTemplate \*addressTmpl)=0;

//Called by CH to log the check any operation

virtual void tliCheckedAny\_c (const Tstring &am, const timeval ts, const Tstring src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log the mismatch in a check any operation

virtual void tliCheckAnyMismatch\_m (const Tstring &am, const timeval ts, const Tstring &src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TciValue \*addrValue, const TciValueTemplate \*addressTmpl)=0;

//Called by CH to log the mismatch in a check any operation

virtual void tliCheckAnyMismatch\_c (const Tstring &am, const timeval ts, const Tstring src,

const Tinteger line, const TriComponentId \*c, const TriPortId \*at,

const TriComponentId \*from, const TciNonValueTemplate \*fromTmpl)=0;

//Called by TE to log the generation of a random number

virtual void tliRnd (const Tstring &am, const timeval ts, const Tstring src, const Tinteger line, const TriComponentId \*c, const FloatValue \*val, const FloatValue \*seed)=0;

//Called by TE to log evaluation of a @fuzzy or @lazy template or variable

virtual void tliEvaluate (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const QualifiedName &name, const TciValue \*evalResult)=0;

//Called by TE to log the component call operation

virtual void tliCCall (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 end of a component call

virtual void tliCCallTerminated (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const VerdictValue \*verdict, const TString &reason,  
TciParameterList \*tciPars, const TciValue \*returnValue)=0;

//Called by TE to log the start of a parameterized module control function

virtual void tliCtrlStartWithParameters (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const TciParameterList \*tciPars)=0;

//Called by TE to log the termination of a parameterized module control function

virtual void tliCtrlTerminatedWithResult (const Tstring &am, const timeval ts, const Tstring &src, const Tinteger line, const TriComponentId \*c, const TciValue \*val, const TciParameterList \*tciPars)=0;