#### 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.

In addition, Value can be used to represent matching mechanisms, which are used instead or inside values e.g. in template parameters or for template variables. Two additional operations: isMatchingSymbol (returns true for matching symbols) and valueToString (for printing value content in the same way as the log operation; can be used for displaying value content) are defined. These operations are not mandatory - it is up to a tool vendor to support them or not.

Values using @lazy and @fuzzy modifiers are represented by the Value data type too. However, it is not possible to use the Value data type to perform evaluation of these values; evaluation can be performed by the TE only. If a @lazy or @fuzzy value has been assigned, but it doesn’t contain result of the evaluation, any data access operations shall result in an error.

##### 7.2.2.2.1 The abstract data type Value

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

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

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

TString getValueEncoding() Returns the value encoding attribute as defined in the TTCN-3 module, if any. If no encoding attribute is defined the distinct value null is returned.

TString getValueEncodingVariant() Returns the value encoding variant attribute as defined in the TTCN-3 module, if any. If no encoding variant attribute is defined the distinct value null is returned.

TBoolean isMatchingSymbol() Returns true if at least one of the following is true for the specified value and false in all other cases:

- It is one of the matching mechanisms specified in clauses B.1.2, B.1.3.1 or B.1.5 of ES 201 873‑1 [1]

- It is a string value containing a matching mechanism specified in clauses B.1.3.1 or B.1.3.2 of ES 201 873‑1 [1]

- It is a record of, set of or array value containing a matching mechanism specified in clauses B.1.3.2 or B.1.3.3 of ES 201 873‑1 [1]

- It is a value that has a matching attribute (specified in clause B.1.4 of ES 201 873‑1 [1]) associated with it

TString valueToString() Returns the same string as produced by the log operation (specified in clause 19.11 of ES 201 873‑1 [1]) with the specified value as its parameter.

TBoolean isFuzzy () Returns true if the specified value has the @fuzzy modifier, false otherwise.

TBoolean isLazy () Returns true if the specified value has the @lazy modifier, false otherwise.

TBoolean isEvaluated () Returns true if the value has been evaluated and its data content is available, false otherwise. In case of uninitialized values, false is always returned. The method is typically used for @lazy values, and it returns false for values that have been assigned, but not evaluated yet and true if the value contains the evaluation result. The method returns false for @fuzzy values, as the result of evaluation is never stored by the TE. For all other values, the method returns true.

##### 7.3.4.1.89 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. If the scope contains modifiers, they prefix the scope string. |
| 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. | |

##### 7.3.4.1.90 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. If the scope contains modifiers, they prefix the scope string. |
| Return Value | void | |
| Constraint | Shall be called by TE to log the leaving of a scope. This event occurs after the scoped 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. | |

##### 7.3.4.1.91 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. In case of @lazy variables, it is called also after performing evaluation as evaluation result is automatically assigned to the variable. | |
| 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. | |

##### 7.3.4.1.121 tliEvaluate

|  |  |  |
| --- | --- | --- |
| Signature | void tliEvaluate (in TString am, in TInteger ts, in TString src,  in TInteger line, in TriComponentIdType c,  in QualifiedName name, in Value evalResult) | |
| 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 template or variable. |
| evalResult | The result of evaluation. |
| Return Value | void | |
| Constraint | Shall be called by TE to log the result of evaluatin of a @lazy or @fuzzy template or variable. This event occurs after the template or variable has been evaluated. | |
| 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. | |

#### 8.3.4.1 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();

public boolean isMatchingSymbol() ;

public String valueToString () ;

public boolean isLazy() ;

public boolean isFuzzy() ;

public boolean isEvaluated() ;

}

**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 the TTCN-3module, if any. If no encoding attribute has been defined the distinct valuenull 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.
* isMatchingSymbol Returns true if the specified value is a matching symbol (see clause 7.2.2.2.1 for more details), false otherwise.
* valueToString Returns the same string as produced by the log operation with the specified value as its parameter.
* isLazy Returns true if the specified value is @lazy, false otherwise.
* isFuzzy Returns true if the specified value is @fuzzy, false otherwise.
* isEvaluated Returns true if the specified value contains an evaluation result, false otherwise (see clause 7.2.2.2.1 for more details).

#### 8.5.4.1 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, TciParameterList tciPars, TriTimerDuration dur);

public void tliTcStart(String am, int ts, String src, int line, TriComponentId c,

TciTestCaseId tcId, TciParameterList tciPars, TriTimerDuration dur);

public void tliTcStop(String am, int ts, String src, int line, TriComponentId c, String reason);

public void tliTcStarted(String am, int ts, String src, int line, TriComponentId c,

TciTestCaseId tcId, TciParameterList tciPars, TriTimerDuration dur);

public void tliTcTerminated(String am, int ts, String src, int line, TriComponentId c,

TciTestCaseId tcId, TciParameterList tciPars, VerdictValue verdict, String reason);

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, 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, TciParameterList tciPars,

Value replValue);

public void tliPrGetReplyMismatch\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

TciParameterList tciPars, 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, 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, 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, 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 excTmpl, TciValueDifferenceList diffs,

Value addrValue, TciValueTemplate addressTmpl);

public void tliPrCatchMismatch\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs,

TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliPrCatch\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl,

Value addrValue, TciValueTemplate addressTmpl);

public void tliPrCatch\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl,

TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliPrCatchTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature);

public void tliPrCatchTimeout(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature);

public void tliCCreate(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, String name, Boolean alive);

public void tliCStart(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, TciBehaviourId name, TciParameterList tciPars);

public void tliCRunning(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, ComponentStatus status);

public void tliCAlive(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, ComponentStatus status);

public void tliCStop(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp);

public void tliCKill(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp);

public void tliCDoneMismatch(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, TciNonValueTemplate compTmpl);

public void tliCDone(String am, int ts, String src, int line, TriComponentId c,

TciNonValueTemplate compTmpl);

public void tliCKilledMismatch(String am, int ts, String src, int line, TriComponentId c,

TriComponentId comp, TciNonValueTemplate compTmpl);

public void tliCKilled(String am, int ts, String src, int line, TriComponentId c,

TciNonValueTemplate compTmpl);

public void tliCTerminated(String am, int ts, String src, int line, TriComponentId c,

VerdictValue verdict, String reason);

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 tliPMapParam(String am, int ts, String src, int line, TriComponentId c,

TriPortId port1, TriPortId port2, TciParameterList tciPars,

TciStatus encoderFailure, TriParameterList triPars);

public void tliPUnmap(String am, int ts, String src, int line, TriComponentId c,

TriPortId port1, TriPortId port2);

public void tliPUnmapParam(String am, int ts, String src, int line, TriComponentId c,

TriPortId port1, TriPortId port2, TciParameterList tciPars,

TciStatus encoderFailure, TriParameterList triPars);

public void tliPClear(String am, int ts, String src, int line, TriComponentId c,

TriPortId port);

public void tliPStart(String am, int ts, String src, int line, TriComponentId c,

TriPortId port);

public void tliPStop(String am, int ts, String src, int line, TriComponentId c,

TriPortId port);

public void tliPHalt(String am, int ts, String src, int line, TriComponentId c,

TriPortId port);

public void tliEncode(String am, int ts, String src, int line, TriComponentId c,

Value val, TciStatus encoderFailure, TriMessage msg, String codec);

public void tliDecode(String am, int ts, String src, int line, TriComponentId c,

TriMessage msg, TciStatus decoderFailure, Value val, String codec);

public void tliTTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer);

public void tliTTimeoutMismatch(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TciNonValueTemplate timerTmpl);

public void tliTTimeout(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TciNonValueTemplate timerTmpl);

public void tliTStart(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TriTimerDuration dur);

public void tliTStop(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TriTimerDuration dur);

public void tliTRead(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TriTimerDuration elapsed);

public void tliTRunning(String am, int ts, String src, int line, TriComponentId c,

TriTimerId timer, TimerStatus status);

public void tliSEnter(String am, int ts, String src, int line, TriComponentId c,

QualifiedName name, TciParameterList tciPars, String kind);

public void tliSLeave(String am, int ts, String src, int line, TriComponentId c,

QualifiedName name, 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,

String log);

public void tliAEnter(String am, int ts, String src, int line, TriComponentId c);

public void tliALeave(String am, int ts, String src, int line, TriComponentId c);

public void tliADefaults(String am, int ts, String src, int line, TriComponentId c);

public void tliAActivate(String am, int ts, String src, int line, TriComponentId c,

QualifiedName name, TciParameterList tciPars, Value ref);

public void tliADeactivate(String am, int ts, String src, int line, TriComponentId c,

Value ref);

public void tliANomatch(String am, int ts, String src, int line, TriComponentId c);

public void tliARepeat(String am, int ts, String src, int line, TriComponentId c);

public void tliAWait(String am, int ts, String src, int line, TriComponentId c);

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

public void tliMatch(String am, int ts, String src, int line, TriComponentId c, Value expr,

TciValueTemplate tmpl);

public void tliMatchMismatch(String am, int ts, String src, int line, TriComponentId c,

Value expr, TciValueTemplate tmpl, TciValueDifferenceList diffs);

public void tliInfo (String am, int ts, String src, int line, TriComponentId c,

int level, String info)

public void tliMChecked\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, Value msgValue, TciValueTemplate msgTmpl,

Value addrValue, TciValueTemplate addressTmpl);

public void tliMChecked\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, Value msgValue, TciValueTemplate msgTmpl,

TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliPrGetCallChecked\_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 tliPrGetCallChecked\_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 tliPrGetReplyChecked\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

TciParameterList tciPars, TciValueTemplate parsTmpl,

Value replValue, TciValueTemplate replyTmpl,

Value addrValue, TciValueTemplate addressTmpl);

public void tliPrGetReplyChecked\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

TciParameterList tciPars, TciValueTemplate parsTmpl,

Value replValue, TciValueTemplate replyTmpl,

TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliPrCatchChecked\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl,

Value addrValue, TciValueTemplate addressTmpl);

public void tliPrCatchChecked\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriPortId from, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl,

TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliCheckedAny\_m(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, Value addrValue, TciValueTemplate addressTmpl);

public void tliCheckedAny\_c(String am, int ts, String src, int line, TriComponentId c,

TriPortId at, TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliCheckAnyMismatch\_m(String am, int ts, String src, int line,

TriComponentId c, TriPortId at, Value addrValue, TciValueTemplate addressTmpl);

public void tliCheckAnyMismatch\_c(String am, int ts, String src, int line,

TriComponentId c, TriPortId at, TriComponentId from, TciNonValueTemplate fromTmpl);

public void tliRnd(String am, int ts, String src, int line,

TriComponentId c, FloatValue val, FloatValue seed);

public void tliEvaluate(String am, int ts, String src, int line, TriComponentId c,

QualifiedName name, Value evalResult);

}

## 9.2 Value interfaces

| TCI IDL Interface | ANSI C representation | Notes and comments |
| --- | --- | --- |
| Type | | |
| TciModuleIdType getDefiningModule() | TciModuleIdType  tciGetDefiningModule(Type inst) |  |
| Tstring getName() | String tciGetName(Type inst) | String type reused from IDL (OMG recommendation) |
| TciTypeClassType getTypeClass() | TciTypeClassType tciGetTypeClass  (Type inst) |  |
| Value newInstance() | Value tciNewInstance(Type inst) |  |
| Tstring getTypeEncoding() | String tciGetTypeEncoding(Type inst) |  |
| TstringSeq getTypeExtension() | String\* getTypeExtension(Type inst) |  |
| Tstring getTypeEncodingVariant() | String tciGetTypeEncodingVariant(Type inst) |  |
| Value parseValue(TString val) | Value tciParseValue(Type inst, String val) |  |
| Value | | |
| Tstring getValueEncoding() | String tciGetValueEncoding(Value inst) |  |
| Tstring getValueEncodingVariant() | String tciGetValueEncodingVariant(Value inst) |  |
| Type getType() | Type tciGetType(Value inst) |  |
| Tboolean notPresent() | Boolean tciNotPresent(Value inst) | Boolean type reused from IDL (OMG recommendation). |
|  | void tciSetNull(Value inst) | For optional parameters of operations,  see clause 9.7. |
|  | Boolean tciIsNull(Value inst) | For optional parameters of operations,  see clause 9.7. Boolean type reused from IDL (OMG recommendation). |
| Tboolean isMatchingSymbol() | Boolean tciIsMatchingSymbol(Value inst) |  |
| TString valueToString() | String valueToString(Value inst) |  |
| Tboolean isLazy () | Boolean tciIsLazy(Value inst) |  |
| Tboolean isFuzzy () | Boolean tciIsFuzzy(Value inst) |  |
| Tboolean isEvaluated() | Boolean tciIsEvaluated(Value inst) |  |

#### 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, String reason)

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, String reason)

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 srcint line, TriComponentId c, TriPortId at, TriPortIdList to,

TriSignatureId signature, TciParameterListType tciPars, TriStatus transmissionFailure)

void tliPrCall\_c\_MC

(String am, int ts, String srcint 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 parsTmpl, Value replValue,

TciValueTemplate replyTmpl, TciValueDifferenceList diffs, Value addrValue,

TciValueTemplate addressTmpl)

void tliPrGetReplyMismatch\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl, Value replValue,

TciValueTemplate replyTmpl, TciValueDifferenceList diffs, TriComponentId from,

TciNonValueTemplate fromTmpl)

void tliPrGetReply\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl, Value replValue,

TciValueTemplate replyTmpl, Value addrValue, TciValueTemplate addressTmpl)

void tliPrGetReply\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl, Value replValue,

TciValueTemplate replyTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)

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, String reason)

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 tliPMapParam

(String am, int ts, String src, int line, TriComponentId c, TriPortId port1, TriPortId port2,

TciParameterListType tciPars, TciStatus encoderFailure, TriParameterList triPars)

void tliPUnmap

(String am, int ts, String src, int line, TriComponentId c, TriPortId port1,

TriPortId port2)

void tliPUnmapParam

(String am, int ts, String src, int line, TriComponentId c, TriPortId port1,

TriPortId port2, TciParameterListType tciPars, TciStatus encoderFailure, TriParameterList triPars)

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 tliTTimeoutDetected

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

void tliTTimeoutMismatch

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

TciNonValueTemplate timerTmpl)

void tliTTimeout

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

TciNonValueTemplate timerTmpl)

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 tmpl)

void tliMatchMismatch

(String am, int ts, String src, int line, TriComponentId c, Value expr, TciValueTemplate tmpl,

TciValueDifferenceList diffs);

void tliInfo

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

void tliMChecked\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,

TciValueTemplate msgTmpl, Value addrValue, TciValueTemplate addressTmpl)

void tliMChecked\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, Value msgValue,

TciValueTemplate msgTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)

void tliPrGetCallChecked\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl,

Value addrValue, TciValueTemplate addressTmpl)

void tliPrGetCallChecked\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl,

TriComponentId from, TciNonValueTemplate fromTmpl)

void tliPrGetReplyChecked\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl,

Value replValue, TciValueTemplate replyTmpl, Value addrValue, TciValueTemplate addressTmpl)

void tliPrGetReplyChecked\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

TciParameterListType tciPars, TciValueTemplate parsTmpl,

Value replValue, TciValueTemplate replyTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)

void tliPrCatchChecked\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl, Value addrValue, TciValueTemplate addressTmpl)

void tliPrCatchChecked\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at, TriSignatureId signature,

Value excValue, TciValueTemplate excTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)

void tliCheckedAny\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at,

Value addrValue, TciValueTemplate addressTmpl)

void tliCheckedAny\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at,

TriComponentId from, TciNonValueTemplate fromTmpl)

void tliCheckAnyMismatch\_m

(String am, int ts, String src, int line, TriComponentId c, TriPortId at,

Value addrValue, TciValueTemplate addressTmpl)

void tliCheckAnyMismatch\_c

(String am, int ts, String src, int line, TriComponentId c, TriPortId at,

TriComponentId from, TciNonValueTemplate fromTmpl)

void tliRnd

(String am, int ts, String src, int line, TriComponentId c, TFloat val, TFloat seed)

void tliEvaluate

(String am, int ts, String src, int line, TriComponentId c, QualifiedName name, Value evalResult)

#### 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 isMatchingSymbol () const =0;

virtual const Tstring & valueToString () const =0;

virtual Tboolean isLazy () const =0;

virtual Tboolean isFuzzy () const =0;

virtual Tboolean isEvaluated () const =0;

virtual Tboolean operator== (const TciValue &val) const =0;

virtual TciValue \* clone () 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 the TTCN-3 module

getValueEncodingVariant

Returns the value encoding variant attribute as defined in the TTCN-3 module

notPresent

Returns true if the specified value is omit

isMatchingSymbol

Returns true if the specified value is a matching symbol (see 7.2.2.2.1 for more details), false otherwise

valueToString

Returns the same string as produced by the log operation

isLazy

Returns true if the specified value is @lazy, false otherwise

isFuzzy

Returns true if the specified value is @fuzzy, false otherwise

isEvaluated

Returns true if the specified value contains an evaluation result, false otherwise (see clause 7.2.2.2.1 for more details)

operator==

Returns true if both objects are equal

clone

Return a copy of the TciValue

operator<

Operator < overload

#### 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)=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)=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 TriAddress \*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 TriAddress \*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 TriAddress \*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 TriAddress \*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 TriAddress \*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;

### 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:element name="component" type="Values:ComponentValue"/>

<xsd:element name="port" type="Values:PortValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

<xsd:element name="timer" type="Values:TimerValue"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:simpleType name="ValueModifier">

<xs:restriction base="SimpleTypes:TString">  
      <xs:enumeration value="lazy"/>  
      <xs:enumeration value="fuzzy"/>

    </xs:restriction>

</xsd:simpleType>

<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="modifier" type="Values:ValueModifier" 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.
* component A component value.
* port A port value.
* default A default value.
* timer A timer 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.
* modifier The value modifier, if used: either lazy or fuzzy.
* 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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The integer value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The float value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The boolean value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The verdict value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

</xsd:complexType>

**Choice of Elements:**

* value The bitstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The hexstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The octetstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The charstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:choice >

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The universal charstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:element name="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

</xsd:choice>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* default A default value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* default A default value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:element name="address" type="Values:AddressValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="port" type="Values:PortValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="timer" type="Values:TimerValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* port A port value.
* default A default value.
* timer A timer value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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: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="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

</xsd:choice>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* default A default value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol "/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* default A default value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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="intValue" type="SimpleTypes:TInteger" minOccurs="0"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol "/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

</xsd:sequence>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Sequence of Elements:**

* value The enumeration value.
* intValue The integer value.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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:element name="anytype" type="Values:AnytypeValue"/>

<xsd:element name="address" type="Values:AddressValue"/>

<xsd:element name="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* component A component value.
* default A default value.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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>

<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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**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>

<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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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.
* null If no field is given.
* omit If the field is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**Attributes:**

* The same attributes as those of Value.

#### 11.3.3.21 ComponentValue

Value type used for component instances is mapped to the complex type specified below. The content of the XML elements based on the ComponentValue type shall be equal to the string produced by the valueToString operation (described in clause 7.2.2.2.1):

<xsd:complexType name="ComponentValue">

<xsd:choice>

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The universal charstring value as string.
* null If no value is given.
* omit If the value is omitted.
* matching\_symbol If the value contains matching symbols.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**Attributes:**

* The same attributes as those of Value.

#### 11.3.3.22 PortValue

Value type used for port instances is mapped to the complex type specified below. The content of the XML elements based on the PortValue type shall be equal to the string produced by the valueToString operation (described in clause 7.2.2.2.1):

<xsd:complexType name="PortValue">

<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>

**Choice of Elements:**

* 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.

#### 11.3.3.23 DefaultValue

Value type used for default instances is mapped to the complex type specified below. The content of the XML elements based on the DefaultValue type shall be equal to the string produced by the valueToString operation (described in clause 7.2.2.2.1):

<xsd:complexType name="DefaultValue">

<xsd:choice>

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

**Choice of Elements:**

* value The universal charstring value as string.
* null If no value is given.
* omit If the value is omitted.
* not\_evaluated Used if a @lazy or @fuzzy value contains not evaluated content.

**Attributes:**

* The same attributes as those of Value.

#### 11.3.3.24 TimerValue

Value type used for timer instances is mapped to the complex type specified below. The content of the XML elements based on the TimerValue type shall be equal to the string produced by the valueToString operation (described in clause 7.2.2.2.1):

<xsd:complexType name="TimerValue">

<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>

**Choice of Elements:**

* 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.

#### 11.4.2.1 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:sequence>

<xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</xsd:sequence>

</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:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</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:complexType name="tliMSend\_c">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMSend\_c\_BC">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMSend\_c\_MC">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMDetected\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Types:TriMessageType"/>

<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMDetected\_c">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMMismatch\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>

<xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>

<xsd:element name="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: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: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="parsTmpl" 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="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="parsTmpl" 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: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="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: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: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 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: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="excTmpl" 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="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="hostId" type="Values:Value" minOccurs="0"/>

<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="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCKilledMismatch">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="comp" type="Types:TriComponentIdType"/>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCDone">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCKilled">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCTerminated">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="verdict" type="Values:VerdictValue" />

<xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</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="tliPMapParam">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:tliPMap">

<xsd:sequence>

<xsd:element name="tciPars" type="Types:TciParameterListType" />

<xsd:choice>

<xsd:element name="encoder‑failure" type="SimpleTypes:TciStatusType" minOccurs="0"/>

<xsd:element name="triPars" type="Types:TriParameterListType" />

</xsd:choice>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliPUnmap">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:PortConfiguration"/>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliPUnmapParam">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:tliPUnmap">

<xsd:sequence>

<xsd:element name="tciPars" type="Types:TciParameterListType" />

<xsd:choice>

<xsd:element name="encoder‑failure" type="SimpleTypes:TciStatusType" minOccurs="0"/>

<xsd:element name="triPars" type="Types:TriParameterListType"/>

</xsd:choice>

</xsd:sequence>

</xsd:extension>

</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="tliTTimeoutDetected">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTTimeoutMismatch">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

<xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTTimeout">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

<xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTStart">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTStop">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTRead">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="elapsed" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTRunning">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="status" type="SimpleTypes:TimerStatusType"/>

</xsd:sequence>

</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: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>

<xsd:complexType name="tliMChecked\_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="tliMChecked\_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="tliPrGetCallChecked\_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="tliPrGetCallChecked\_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="tliPrGetReplyChecked\_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="tliPrGetReplyChecked\_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="tliPrCatchChecked\_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="tliPrCatchChecked\_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="tliCheckedAny\_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="address" type="Types:TriAddressType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCheckedAny\_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:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCheckMismatch\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<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="tliCheckMismatch\_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:TriComponentIdType"

minOccurs="0"/>

<xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"

minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliRnd">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="val" type="Values:FloatValue"/>

<xsd:element name="seed" type="Values:FloatValue"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliEvaluate">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="name" type="Types:QualifiedName" />

<xsd:element name="evalResult" type="Values:Value" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

#### 12.4.4.1 Value

The IDL type **Value** is mapped to the following interface:

public interface ITciValue {  
 ITciType Type { get; }  
 bool NotPresent { get; }  
 string ValueEncoding { get; }  
 string ValueEncodingVariant { get; }  
 bool IsMatchingSymbol { get; }  
 string ValueToString();  
 bool IsLazy { get; }  
 bool IsFuzzy { get; }  
 bool IsEvaluated { get; }  
}

**Members:**

* Type  
  Returns the type of the specified value.
* NotPresent  
  Returns true if the specified value is omit, false otherwise.
* ValueEncoding  
  This property 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.
* ValueEncodingVariant  
  This property 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.
* NotPresent  
  Returns true if the specified value is a matching symbol (see clause 7.2.2.2.1 for more details), false otherwise.
* ValueToString  
  Returns the same string as produced by the log operation with the specified value as its parameter.
* IsLazy  
  Returns true if the specified value is @lazy, false otherwise.
* IsFuzzy  
  Returns true if the specified value is @fuzzy, false otherwise.
* IsEvaluated  
  Returns true if the specified value contains an evaluation result, false otherwise (see clause 7.2.2.2.1 for more details).

#### 12.5.4.1 TCI-TL provided

The **TCI-TL provided** interface is mapped to the following interface:

public interface ITciTLProvided {  
 void TliTcExecute(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciTestCaseId tcId,  
 ITciParameterList tciPars, ITriTimerDuration dur);  
 void TliTcStart(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciTestCaseId tcId,  
 ITciParameterList tciPars, ITriTimerDuration dur);  
 void TliTcStop(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, string reason);  
 void TliTcStarted(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciTestCaseId tcId,  
 ITciParameterList tciPars, ITriTimerDuration dur);  
 void TliTcTerminated(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciTestCaseId tcId,  
 ITciParameterList tciPars, ITciVerdictValue verdict,  
 string reason);  
 void TliCtrlStart(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliCtrlStop(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliCtrlTerminated(string am, System.DateTime ts, string src, int  
 line, ITriComponentId c);  
 void TliMSend\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to, ITciValue msgValue,  
 ITciValue addrValue, TciStatus encoderFailure,  
 ITriMessage msg, ITriAddress address,  
 TriStatus transmissionFailure);  
 void TliMSend\_m\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to, ITciValue msgValue,  
 TciStatus encoderFailure, ITriMessage msg,  
 TriStatus transmissionFailure);  
 void TliMSend\_m\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to, ITciValue msgValue,  
 ITciValueList addrValues, TciStatus encoderFailure,  
 ITriMessage msg,  
 ITriAddressList addresses,  
 TriStatus transmissionFailure);  
 void TliMSend\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to, ITciValue msgValue,  
 TriStatus transmissionFailure);  
 void TliMSend\_c\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to, ITciValue msgValue,  
 TriStatus transmissionFailure);  
 void TliMSend\_c\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to, ITciValue msgValue,  
 TriStatus transmissionFailure);  
 void TliMDetected\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId from, ITriMessage msg,  
 ITriAddress address);  
 void TliMDetected\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId from, ITciValue msgValue);  
 void TliMMismatch\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl,  
 ITciValueDifferenceList diffs, ITciValue address,  
 ITciValueTemplate addressTmpl);  
 void TliMMismatch\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl,  
 ITciValueDifferenceList diffs, ITriComponentId from,  
 ITciNonValueTemplate fromTmpl);  
 void TliMReceive\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl, ITciValue address,  
 ITciValueTemplate addressTmpl);  
 void TliMReceive\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl,  
 ITriComponentId fromComp,  
 ITciNonValueTemplate fromTmpl);  
 void TliPrCall\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValue addrValue,  
 TciStatus encoderFailure, ITriParameterList triPars,  
 ITriAddress address,  
 TriStatus transmissionFailure);  
 void TliPrCall\_m\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 TciStatus encoderFailure, ITriParameterList triPars,  
 TriStatus transmissionFailure);  
 void TliPrCall\_m\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValueList addrValues, TciStatus encoderFailure,  
 ITriParameterList triPars,  
 ITriAddressList addresses,  
 TriStatus transmissionFailure);  
 void TliPrCall\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature,  
 ITciParameterList tciPars,  
 TriStatus transmissionFailure);  
 void TliPrCall\_c\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 TriStatus transmissionFailure);  
 void TliPrCall\_c\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 TriStatus transmissionFailure);  
 void TliPrGetCallDetected\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature,  
 ITriParameterList triPars,  
 ITriAddress address);  
 void TliPrGetCallDetected\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature,  
 ITciParameterList tciPars);  
 void TliPrGetCallMismatch\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValueTemplate parsTmpl, ITciValueDifferenceList diffs,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrGetCallMismatch\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITciValueDifferenceList diffs, ITriComponentId from,  
 ITciNonValueTemplate fromTmpl);  
 void TliPrGetCall\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrGetCall\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrReply\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, ITciValue addrValue, TciStatus encoderFailure,  
 ITriParameterList triPars,  
 ITriParameter repl,  
 ITriAddress address,  
 TriStatus transmissionFailure);  
 void TliPrReply\_m\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, TciStatus encoderFailure,  
 ITriParameterList triPars,  
 ITriParameter repl,  
 TriStatus transmissionFailure);  
 void TliPrReply\_m\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, ITciValueList addrValues,  
 TciStatus encoderFailure, ITriParameterList triPars,  
 ITriParameter repl,  
 ITriAddressList addresses,  
 TriStatus transmissionFailure);  
 void TliPrReply\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, TriStatus transmissionFailure);  
 void TliPrReply\_c\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, TriStatus transmissionFailure);  
 void TliPrReply\_c\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue, TriStatus transmissionFailure);  
 void TliPrGetReplyDetected\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature,  
 ITriParameterList triPars,  
 ITriParameter repl,  
 ITriAddress address);  
 void TliPrGetReplyDetected\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue replValue);  
 void TliPrGetReplyMismatch\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValueTemplate parsTmpl, ITciValue replValue,  
 ITciValueTemplate replyTmpl, ITciValueDifferenceList diffs,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrGetReplyMismatch\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValueTemplate parsTmpl, ITciValue replValue,  
 ITciValueTemplate replyTmpl, ITciValueDifferenceList diffs,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrGetReply\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValueTemplate parsTmpl, ITciValue replValue,  
 ITciValueTemplate replyTmpl, ITciValue address,  
 ITciValueTemplate addressTmpl);  
 void TliPrGetReply\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITciValue replValue, ITciValueTemplate replyTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrRaise\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue excValue, ITciValue addrValue, TciStatus encoderFailure,  
 ITriException exc,  
 ITriAddress address,  
 TriStatus transmissionFailure);  
 void TliPrRaise\_m\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue excValue, TciStatus encoderFailure,  
 ITriException exc,  
 TriStatus transmissionFailure);  
 void TliPrRaise\_m\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue excValue, ITciValueList addrValues,  
 TciStatus encoderFailure, ITriException exc,  
 ITriAddressList addresses,  
 TriStatus transmissionFailure);  
 void TliPrRaise\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortId to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue excValue, TriStatus transmissionFailure);  
 void TliPrRaise\_c\_BC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValue excValue,  
 TriStatus transmissionFailure);  
 void TliPrRaise\_c\_MC(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriPortIdList to,  
 ITriSignatureId signature, ITciParameterList tciPars,  
 ITciValue excValue, TriStatus transmissionFailure);  
 void TliPrCatchDetected\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature,  
 ITriException exc,  
 ITriAddress address);  
 void TliPrCatchDetected\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriPortId from,  
 ITriSignatureId signature, ITciValue excValue);  
 void TliPrCatchMismatch\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature, ITciValue excValue,  
 ITciValueTemplate excTmpl, ITciValueDifferenceList diffs,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrCatchMismatch\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature,  
 ITciValue excValue, ITciValueTemplate excTmpl,  
 ITciValueDifferenceList diffs, ITriComponentId from,  
 ITciNonValueTemplate fromTmpl);  
 void TliPrCatch\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature,  
 ITciValue excValue, ITciValueTemplate excTmpl, ITciValue address,  
 ITciValueTemplate addressTmpl);  
 void TliPrCatch\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriSignatureId signature,  
 ITciValue excValue, ITciValueTemplate excTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrCatchTimeoutDetected(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature);  
 void TliPrCatchTimeout(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at,  
 ITriSignatureId signature);  
 void TliCCreate(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp, string name, bool alive);  
 void TliCStart(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp, ITciBehaviourId name,  
 ITciParameterList tciPars);  
 void TliCRunning(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp, TciComponentStatus status);  
 void TliCAlive(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp, TciComponentStatus status);  
 void TliCStop(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,   
 ITriComponentId comp);  
 void TliCKill(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp);  
 void TliCDoneMismatch(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 ITriComponentId comp, ITciNonValueTemplate compTmpl);  
 void TliCDone(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciNonValueTemplate compTmpl);  
 void TliCKilledMismatch(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriComponentId comp, ITciNonValueTemplate compTmpl);  
 void TliCKilled(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciNonValueTemplate compTmpl);  
 void TliCTerminated(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciVerdictValue verdict,  
 string reason);  
 void TliPConnect(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port1,  
 ITriPortId port2);  
 void TliPDisconnect(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port1,  
 ITriPortId port2);  
 void TliPMap(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port1,  
 ITriPortId port2);  
 void TliPUnmap(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port1,  
 ITriPortId port2);  
 void TliPClear(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port);  
 void TliPStart(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port);  
 void TliPStop(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port);  
 void TliPHalt(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId port);  
 void TliEncode(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciValue val,  
 TciStatus encoderFailure, ITriMessage msg,  
 string codec);  
 void TliDecode(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriMessage msg,  
 TciStatus decoderFailure, ITciValue val, string codec);  
 void TliTTimeoutDetected(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriTimerId timer);  
 void TliTTimeoutMismatch(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriTimerId timer, ITciNonValueTemplate timerTmpl);  
 void TliTTimeout(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriTimerId timer,  
 ITciNonValueTemplate timerTmpl);  
 void TliTStart(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriTimerId timer,  
 ITriTimerDuration dur);  
 void TliTStop(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriTimerId timer,  
 ITriTimerDuration dur);  
 void TliTRead(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriTimerId timer,  
 ITriTimerDuration elapsed);  
 void TliTRunning(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriTimerId timer,  
 TciTimerStatus status);  
 void TliSEnter(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciParameterList tciPars,  
 string kind);  
 void TliSLeave(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciParameterList tciPars,  
 ITciValue returnValue, string kind);  
 void TliVar(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciValue varValue);  
 void TliModulePar(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciValue parValue);  
 void TliGetVerdict(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciVerdictValue verdict);  
 void TliSetVerdict(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciVerdictValue verdict,  
 string reason);  
 void TliLog(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, string log);  
 void TliAEnter(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliALeave(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliADefaults(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliAActivate(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciParameterList tciPars,  
 ITciValue expr);  
 void TliADeactivate(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciValue expr);  
 void TliANomatch(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliARepeat(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliAWait(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c);  
 void TliAction(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, string action);  
 void TliMatch(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciValue expr,  
 ITciValueTemplate tmpl);  
 void TliMatchMismatch(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITciValue expr,  
 ITciValueTemplate tmpl, ITciValueDifferenceList diffs);  
 void TliInfo (string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, int level, string info);  
 void TliMChecked\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl,

ITciValue address, ITciValueTemplate addressTmpl);  
 void TliMChecked\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue msgValue, ITciValueTemplate msgTmpl,  
 ITriComponentId fromComp, ITciNonValueTemplate fromTmpl);  
 void TliPrGetCallChecked\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrGetCallChecked\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrGetReplyChecked\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,  
 ITciParameterList tciPars, ITciValueTemplate parsTmpl,

ITciValue replValue, ITciValueTemplate replyTmpl,

ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrGetReplyChecked\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,

ITciParameterList tciPars, ITciValueTemplate parsTmpl,   
 ITciValue replValue, ITciValueTemplate replyTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);  
 void TliPrCatchChecked\_m(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,  
 ITciValue excValue, ITciValueTemplate excTmpl,

ITciValue address, ITciValueTemplate addressTmpl);  
 void TliPrCatchChecked\_c(string am, System.DateTime ts, string src,  
 int line, ITriComponentId c,  
 ITriPortId at, ITriSignatureId signature,

ITciValue excValue, ITciValueTemplate excTmpl,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);

void TliCheckedAny\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITciValue address, ITciValueTemplate addressTmpl);  
 void TliCheckedAny\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,  
 ITriComponentId from, ITciNonValueTemplate fromTmpl);

void TliCheckAnyMismatch\_m(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,

ITciValue address, ITciValueTemplate addressTmpl);  
 void TliCheckAnyMismatch\_c(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, ITriPortId at,

ITriComponentId from, ITciNonValueTemplate fromTmpl);

void TliRnd(string am, System.DateTime ts, string src, int line,  
 ITriComponentId c, FloatValue val, FloatValue seed);

void TliEvaluate (string am, System.DateTime ts, string src, int line,  
 ITriComponentId c,  
 IQualifiedName name, ITciValue evalResult);  
}

# 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\_v4\_6\_1.xsd"

xmlns:Values="http://uri.etsi.org/ttcn‑3/tci/Values\_v4\_6\_1.xsd"

xmlns:Templates="http://uri.etsi.org/ttcn‑3/tci/Templates\_v4\_6\_1.xsd"

xmlns:SimpleTypes="http://uri.etsi.org/ttcn‑3/tci/SimpleTypes\_v4\_6\_1.xsd" elementFormDefault="qualified">

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/Templates\_v4\_6\_1.xsd"

schemaLocation="Templates\_v4\_6\_1.xsd"/>

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/SimpleTypes\_v4\_6\_1.xsd"

schemaLocation="SimpleTypes\_v4\_6\_1.xsd"/>

<xsd:simpleType name="ValueModifier">

<xs:restriction base="SimpleTypes:TString">  
      <xs:enumeration value="lazy"/>  
      <xs:enumeration value="fuzzy"/>

    </xs:restriction>

</xsd:simpleType>

<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:attribute name="annotation" type="Values:ValueModifier" 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:element name="component" type="Values:ComponentValue"/>

<xsd:element name="port" type="Values:PortValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

<xsd:element name="timer" type="Values:TimerValue"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="NotEvaluated">

<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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

</xsd:choice>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="port" type="Values:PortValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="timer" type="Values:TimerValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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:element name="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

</xsd:choice>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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="component" type="Values:ComponentValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="default" type="Values:DefaultValue" minOccurs="0"

maxOccurs="unbounded"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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="intValue" type="SimpleTypes:TInteger" minOccurs="0"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</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="component" type="Values:ComponentValue"/>

<xsd:element name="default" type="Values:DefaultValue"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="AnytypeValue">

<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: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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="AddressValue">

<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: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:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="ComponentValue">

<xsd:choice>

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="matching\_symbol" type="Templates:MatchingSymbol"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="PortValue">

<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="DefaultValue">

<xsd:choice>

<xsd:element name="value" type="SimpleTypes:TString"/>

<xsd:element name="null" type="Templates:null"/>

<xsd:element name="omit" type="Templates:omit"/>

<xsd:element name="not\_evaluated" type="Values:NotEvaluated"/>

</xsd:choice>

<xsd:attributeGroup ref="Values:ValueAtts"/>

</xsd:complexType>

<xsd:complexType name="TimerValue">

<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: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\_v4\_6\_1.xsd"

xmlns:Events="http://uri.etsi.org/ttcn‑3/tci/Events\_v4\_6\_1.xsd"

xmlns:Types="http://uri.etsi.org/ttcn‑3/tci/Types\_v4\_6\_1.xsd"

xmlns:Templates="http://uri.etsi.org/ttcn‑3/tci/Templates\_v4\_6\_1.xsd"

xmlns:SimpleTypes="http://uri.etsi.org/ttcn‑3/tci/SimpleTypes\_v4\_6\_1.xsd"

xmlns:Values="http://uri.etsi.org/ttcn‑3/tci/Values\_v4\_6\_1.xsd"

elementFormDefault="qualified">

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/SimpleTypes\_v4\_6\_1.xsd"

schemaLocation="SimpleTypes\_v4\_6\_1.xsd"/>

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/Types\_v4\_6\_1.xsd" schemaLocation="Types\_v4\_6\_1.xsd"/>

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/Values\_v4\_6\_1.xsd"

schemaLocation="Values\_v4\_6\_1.xsd"/>

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/Templates\_v4\_6\_1.xsd"

schemaLocation="Templates\_v4\_6\_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:sequence>

<xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</xsd:sequence>

</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:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</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:complexType name="tliMSend\_c">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMSend\_c\_BC">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMSend\_c\_MC">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="to" type="Types:TriPortIdListType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="transmission‑failure" type="SimpleTypes:TriStatusType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMDetected\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Types:TriMessageType"/>

<xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMDetected\_c">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="from" type="Types:TriPortIdType" minOccurs="0"/>

<xsd:element name="msgValue" type="Values:Value"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliMMismatch\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<xsd:element name="msgValue" type="Values:Value"/>

<xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>

<xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>

<xsd:element name="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: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: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="parsTmpl" 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="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="parsTmpl" 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: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="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: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: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 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: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="excTmpl" 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="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="hostId" type="Values:Value" minOccurs="0"/>

<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="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCKilledMismatch">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="comp" type="Types:TriComponentIdType"/>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCDone">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCKilled">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCTerminated">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="verdict" type="Values:VerdictValue" />

<xsd:element name="reason" type="SimpleTypes:TString" minOccurs="0"/>

</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="tliPMapParam">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:tliPMap">

<xsd:sequence>

<xsd:element name="tciPars" type="Types:TciParameterListType" />

<xsd:choice>

<xsd:element name="encoder‑failure" type="SimpleTypes:TciStatusType" minOccurs="0"/>

<xsd:element name="triPars" type="Types:TriParameterListType" />

</xsd:choice>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliPUnmap">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:PortConfiguration"/>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliPUnmapParam">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:tliPUnmap">

<xsd:sequence>

<xsd:element name="tciPars" type="Types:TciParameterListType" />

<xsd:choice>

<xsd:element name="encoder‑failure" type="SimpleTypes:TciStatusType" minOccurs="0"/>

<xsd:element name="triPars" type="Types:TriParameterListType"/>

</xsd:choice>

</xsd:sequence>

</xsd:extension>

</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="tliTTimeoutDetected">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTTimeoutMismatch">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

<xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTTimeout">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType" />

<xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" />

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTStart">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTStop">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="dur" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTRead">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="elapsed" type="SimpleTypes:TriTimerDurationType"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliTRunning">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="timer" type="Types:TriTimerIdType"/>

<xsd:element name="status" type="SimpleTypes:TimerStatusType"/>

</xsd:sequence>

</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: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>

<xsd:complexType name="tliMChecked\_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="tliMChecked\_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>

<xsd:complexType name="tliPrGetCallChecked\_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="tliPrGetCallChecked\_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="tliPrGetReplyChecked\_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="tliPrGetReplyChecked\_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="tliPrCatchChecked\_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="tliPrCatchChecked\_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="tliCheckedAny\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<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="tliCheckedAny\_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:TriComponentIdType"

minOccurs="0"/>

<xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"

minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliCheckMismatch\_m">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="at" type="Types:TriPortIdType"/>

<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="tliCheckMismatch\_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:TriComponentIdType"

minOccurs="0"/>

<xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"

minOccurs="0"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliRnd">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="val" type="Values:FloatValue"/>

<xsd:element name="from" type="Values:FloatValue"/>

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>

<xsd:complexType name="tliEvaluate">

<xsd:complexContent mixed="true">

<xsd:extension base="Events:Event">

<xsd:sequence>

<xsd:element name="name" type="Types:QualifiedName" />

<xsd:element name="evalResult" type="Values:Value" minOccurs="0"/>

</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\_v4\_6\_1.xsd"

xmlns:TLI="http://uri.etsi.org/ttcn‑3/tci/TLI\_v4\_6\_1.xsd"

xmlns:Events="http://uri.etsi.org/ttcn‑3/tci/Events\_v4\_6\_1.xsd"

elementFormDefault="qualified">

<xsd:import namespace="http://uri.etsi.org/ttcn‑3/tci/Events\_v4\_6\_1.xsd"

schemaLocation="Events\_v4\_6\_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: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="tliPMapParam" type="Events:tliPMapParam"/>

<xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>

<xsd:element name="tliPUnmapParam" type="Events:tliPUnmapParam"/>

<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"/>

<!‑‑ check ‑‑>

<xsd:element name="tliMChecked\_m" type="Events:tliMChecked\_m"/>

<xsd:element name="tliMChecked\_c" type="Events:tliMChecked\_c"/>

<xsd:element name="tliPrGetCallChecked\_m" type="Events:tliPrGetCallChecked\_m"/>

<xsd:element name="tliPrGetCallChecked\_c" type="Events:tliPrGetCallChecked\_c"/>

<xsd:element name="tliPrGetReplyChecked\_m" type="Events:tliPrGetReplyChecked\_m"/>

<xsd:element name="tliPrGetReplyChecked\_c" type="Events:tliPrGetReplyChecked\_c"/>

<xsd:element name="tliPrCatchChecked\_m" type="Events:tliPrCatchChecked\_m"/>

<xsd:element name="tliPrCatchChecked\_c" type="Events:tliPrCatchChecked\_c"/>

<xsd:element name="tliCheckedAny\_m" type="Events:tliCheckedAny\_m"/>

<xsd:element name="tliCheckedAny\_c" type="Events:tliCheckedAny\_c"/>

<xsd:element name="tliCheckAnyMismatch\_m" type="Events:tliCheckAnyMismatch\_m"/>

<xsd:element name="tliCheckAnyMismatch\_c" type="Events:tliCheckAnyMismatch\_c"/>

<!—- rnd -->

<xsd:element name="tliRnd" type="Events:tliRnd"/>

<xsd:element name="tliEvaluate" type="Events:tliEvaluate"/>

</xsd:choice>

</xsd:complexType>

</xsd:schema>