国 际 电 信 联 盟

ITU-T

国际电信联盟
电信标准化部门

Z.145

(03/2006)

Z系列：电信系统使用的语言和一般性软件情况

正式描述技巧（FDT）— 测试和测试控制记法（TTCN）

测试和测试控制记法第三版（TTCN-3）：控制接口（TCI）

ITU-T Z.145建议书

国际电信联盟

ITU-T Z系列建议书

电信系统使用的语言和一般性软件情况

正式描述技巧（FDT）	
规范和描述语言（SDL）	Z.100-Z.109
正式描述技巧的应用	Z.110-Z.119
信息排序表（MSC）	Z.120-Z.129
扩展的目标描述语言（eODL）	Z.130-Z.139
测试和测试控制记法（TTCN）	**Z.140-Z.149**
用户要求记法（URN）	Z.150-Z.159
编程语言	
CHILL: ITU-T 高级语言	Z.200-Z.209
人机语言	
总则	Z.300-Z.309
基本句法和对话程序	Z.310-Z.319
用于视频显示终端的扩展 MML	Z.320-Z.329
人机接口规范	Z.330-Z.349
面向数据的人机接口	Z.350-Z.359
电信网络管理使用的人机接口	Z.360-Z.379
质量	
电信软件的质量	Z.400-Z.409
涉及协议的建议书中有关质量的内容	Z.450-Z.459
方法	
认证与测试的方法	Z.500-Z.519
中间件	
分布或处理环境	Z.600-Z.609

欲了解更详细信息，请查阅*ITU-T*建议书目录。

ITU-T Z.145建议书

测试和测试控制记法第三版（TTCN-3）：
控制接口（TCI）

摘　要

本建议书规定了用于 TTCN-3（*测试和测试控制记法 3*）测试系统实施的控制接口。TTCN-3 控制接口为测试系统到特定测试平台的管理、测试部件的处理和编码/解码，提供建议的适配。本建议书把接口定义为一套与目标语言不相关的操作。

接口的定义与 ITU-T Z.140 建议书兼容。本建议书中接口的定义使用 CORBA 接口定义语言（IDL）完整地规定 TCI。第 8 节和第 9 节规定了抽象规范到目标语言 Java 和 ANSI-C 的语言映射。基于 IDL 的接口规范在附件 A 中概述。

来　源

ITU-T 第 17 研究组（2005-2008）按照 ITU-T A.8 建议书规定的程序，于 2006 年 3 月 16 日批准了 ITU-T Z.145 建议书。

前　言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定 ITU-T 各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA 第 1 号决议规定了批准建议书须遵循的程序。

属 ITU-T 研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简明扼要起见而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联尚未收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能并非最新信息，因此特大力提倡他们通过下列网址查询电信标准化局（TSB）的专利数据库：http://www.itu.int/ITU-T/ipr/。

目 录

ITU-T Z.145建议书

测试和测试控制记法第三版（TTCN-3）：控制接口（TCI）

1 范围

本建议书规定了 TTCN-3 测试系统实施方案的控制接口。针对某个特定的测试平台，TTCN-3 控制接口对测试系统的管理、测试部件处理和编码/解码做了标准的改造。本建议书把接口定义为一组独立于目标语言的操作。

定义的接口兼容于 TTCN-3 标准（参见以下参考文献）。接口定义使用 CORBA 接口定义语言（IDL）来彻底规范 TCI。第 8 节和第 9 节描述了本抽象规范与目标语言 Java 和 ANSI-C 之间的语言映射关系。附件 A 对基于 IDL 的接口规范做了概述。

2 参考文献

下列 ITU-T 建议书和其它参考文献的条款，通过在本建议书中的引用而构成本建议书的条款。在出版时，所指出的版本是有效的。所有的建议书和其它参考文献都面临修订，使用本建议书的各方应探讨使用下列建议书和其它参考文献最新版本的可能性。当前有效的 ITU-T 建议书清单定期出版。本建议书中引用某个独立文件，并非确定该文件具备建议书的地位。

[1] ITU-T Recommendation Z.144 (2006), *Testing and Test Control Notation version 3 (TTCN-3): Runtime interface (TRI).*

[2] ITU-T Recommendation Z.140 (2006), *Testing and Test Control Notation version 3 (TTCN-3): Core language*.

[3] ITU-T Recommendation Z.143 (2006), *Testing and Test Control Notation version 3 (TTCN-3) Operational semantics*.

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

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

[5] W3C Recommendation (2004), *XML Schema Part 0: Primer*.

注 一 参见：http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[6] W3C Recommendation (2004), *XML Schema Part 1: Structures*.

注 一 参见：http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[7] W3C Recommendation (2004), *XML Schema Part 2: Datatypes*.

注 一 参见：http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

3 定义和缩写

3.1 定义

ITU-T X.290 建议书[4]中和下面给出的术语和定义适用于本建议书。

3.1.1 Abstract Test Suite (ATS) 抽象测试组：由抽象测试用例组成的测试组。

3.1.2 codec 编码解码器：分别用于对发送和接收数据进行编码和解码的编码器/解码器实体。

3.1.3 Coding/Decoding (CD) 编码器/解码器：负责管理值和类型处理的实体，包括 TTCN-3 测试系统中的编码和解码。

3.1.4 Component Handling (CH) 部件处理器：负责管理 TTCN-3 测试系统中测试部件处理的实体。

3.1.5 communication port 通信端口：以利测试部件间通信的抽象机制。

注 — 在接收方向上，一个通信端口建模为一个FIFO队列。端口可以是基于消息的、基于过程的或二者的混合。

3.1.6 control component 控制部件：执行 TTCN-3 模块中控制部分行为的部件。

3.1.7 Executable Test Suite (ETS) 可执行的测试组：指的是 ITU-T X.290 建议书[4]。

3.1.8 Implementation eXtra Information for Testing (IXIT) 有关测试的实施方案额外信息：指的是 ITU-T X.290 建议书[4]。

3.1.9 Platform Adaptor (PA) 平台适配器：使可执行 TTCN-3 适配某个特定执行平台的实体。

注 — 平台适配器为TTCN-3测试系统创建了一个单独的时间概念，并执行显性的和隐性的定时器及外部函数。

3.1.10 real test system interface 实际测试系统接口：指的是 ITU-T X.290 建议书 [4]。

3.1.11 System Under Test (SUT) 接受测试的系统：指的是 ITU-T X.290 建议书[4]。

3.1.12 SUT Adaptor (SA) SUT 适配器：使 TTCN-3 通信操作适配基于抽象测试系统接口之 SUT 的实体。它执行实际测试系统接口。

3.1.13 Testing and Test Control Notation (TTCN-3) 测试和测试控制记法：指的是 ITU-T X.290 建议书[4]。

3.1.14 test case 测试用例：指的是 ITU-T X.290 建议书[4]。

3.1.15 test event 测试事件：通信端口上发送或接收的测试数据（消息或过程调用），是测试系统接口以及定时器超时事件的一部分。

3.1.16 Test Management (TM) 测试管理器：为 TTCN-3 测试系统提供用户接口和管理的实体。

3.1.17 Test Logging (TL) 测试记录器：提供有关测试执行情况的记录信息（也包括 TTCN-3 记录声明提供的信息）。

3.1.18 Test Management and Control (TMC) 测试管理控制器：提供测试管理和控制的实体集；由测试管理器（TM）、部件处理器（CH）、测试记录器（TL）和编码器/解码器（CD）组成。

注 — TMC是TCI的一个实施方案。

3.1.19 test system 测试系统：指的是 ITU-T X.290 建议书[4]。

3.1.20 Test system interface (TSI) 测试系统接口：提供（抽象）TTCN-3 测试系统中可用的端口与实际测试系统提供的端口之间映射的测试部件。

3.1.21 TTCN-3 Executable (TE) 可执行 TTCN-3：测试系统的一部分，用于解释或执行 TTCN-3 ETS。

3.1.22 TTCN-3 Control Interfaces (TCI) TTCN-3 控制接口：定义可执行 TTCN-3 与测试系统中测试管理器、编码器和解码器、测试部件处理器之间交互作用的三个接口。

3.1.23 TTCN-3 Runtime Interface (TRI) TTCN-3 运行时间接口：定义可执行 TTCN-3 与测试系统中 SUT 和平台适配器之间交互作用的接口。

3.2 缩写

本建议书使用以下缩写：

ATS	抽象测试组
CD	编码器/解码器
CH	部件处理器
ETS	可执行的测试组
IDL	接口定义语言
IXIT	有关测试的实施方案额外信息
MSC	消息序列图
MTC	主测试部件
OMG	目标管理小组
PA	平台适配器
PTC	并行测试部件
SA	SUT 适配器
SUT	接受测试的系统
TC	测试控制
TCI	TTCN-3 控制接口
TE	可执行 TTCN-3
TL	测试记录器
TM	测试管理器
TMC	测试管理控制器
TRI	TTCN-3 运行时间接口
TSI	测试系统接口
TTCN-3	测试和测试控制记法第三版

4 引言

本建议书由两个不同部分组成，第一部分描述TTCN-3测试系统实施方案的结构，第二部分描述TTCN-3控制接口规范。

第一部分介绍把 TTCN-3 测试系统分解为四个主要实体：

- 测试管理控制器（TMC）；
- 可执行TTCN-3（TE）；
- SUT适配器（SA）；以及
- 平台适配器（PA）。

TMC 自身由三个实体组成：测试管理器（TM）、编码器/解码器（CD）和测试部件处理器（CH）。另外，定义了这些实体之间的交互作用，即相应的接口。

本建议书的第二部分规范 TTCN-3 控制接口（TCI）。依据作为某个实体一部分而执行并被其它测试系统实体调用的操作来定义接口。对各个操作，接口规范定义了相关的数据结构、对测试系统预期的效果，以及有关操作用法的任何约束条件。注意：这些接口规范只定义 TE 与 TM、TE 与 CD、TE 与 CH 之间的交互作用。对 TE 与 SA、TE 与 PA 之间的交互作用，请参考 TTCN-3 运行时间接口规范（ITU-T Z.144 建议书[1]）。

5 合规性

对符合 TCI 的 TTCN-3 测试系统的最低要求是合乎本建议书中所述的接口规范。测试系统中的 TTCN-3 语义必须合乎 ITU-T Z.143 建议书[3]中所定义的操作语义。另外，必须支持某种语言映射关系。例如，如果某个供货商支持 Java，则作为可执行 TTCN-3 一部分的 TCI 操作调用和执行必须合乎本建议书中规定的 IDL 与 Java 之间的映射关系。对记录接口，可使用 XML 映射，而不使用 Java 或 C 映射。

6 TTCN-3测试系统的一般结构

概念上，一个 TTCN-3 测试系统可认为是各相互作用实体的一个集合。各个实体执行特定的测试系统功能。这些实体：

- 管理测试执行；
- 解释或执行经编译的TTCN-3代码；
- 实现与SUT之间的正确通信；
- 管理类型、值和测试部件；
- 执行外部函数；以及
- 处理定时器操作。

6.1 TTCN-3测试系统中的实体

TTCN-3 测试系统实施方案结构如图 1 所示：

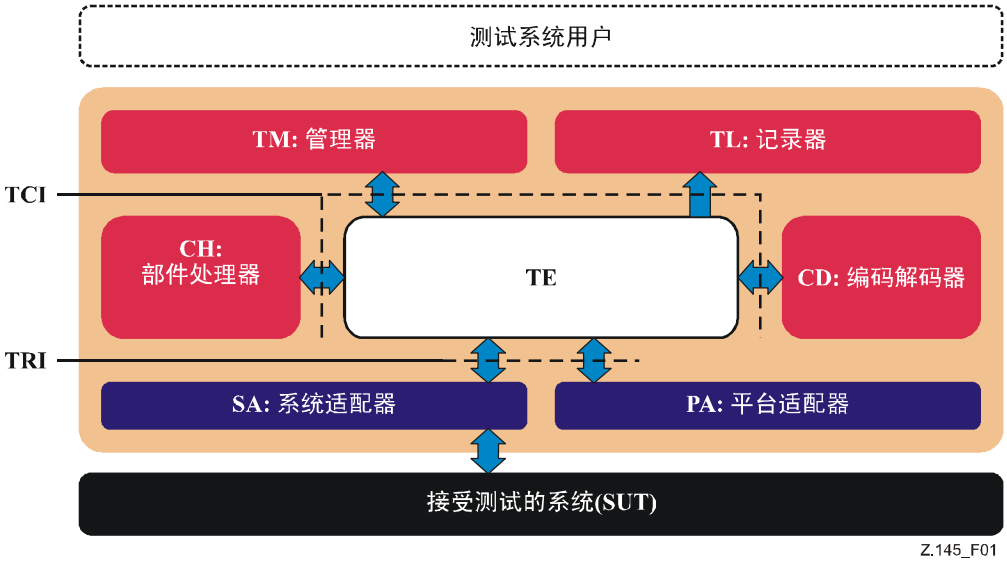

图 1/Z.145－TTCN-3测试系统的一般结构

如图 1 所示，可执行 TTCN-3（TE），即可执行的测试组（ETS），用于解释和执行 TTCN-3 模块。可以确定各种各样的 TE 结构单元：控制、行为、部件、类型、值和队列。TE 内的结构单元表示的是在 TTCN-3 内定义的或由 TTCN-3 标准（ITU-T Z.140 建议书[2]）自身定义的功能。例如，结构单元“控制”表示一个 TTCN-3 模块内的控制部分，而结构单元“队列”表示对一个可执行 TTCN-3 的要求，其测试部件的各个端口负责维护各自的端口队列。前者在一个 TTCN-3 模块内予以规定，后者是 TTCN-3 规范所要求的。

对 TE 做进一步细化，如图 1 所示，用作定义 TTCN-3 控制接口的一种辅助手段。典型地，在一个测试系统实施方案中，TE 对应由 TTCN-3 编译器或 TTCN-3 解释器生成的可执行代码。

可以以集中式或分布式方式来执行 TE，也就是说，分别在一个单个测试设备或在若干个测试设备上来执行。尽管 TE 的结构实体执行一个完整的 TTCN-3 模块，但单个的结构实体可以分布于若干个测试设备上。

TE 在一个抽象层面上执行 TTCN-3 模块。TTCN-3 测试系统的其它实体把这些抽象概念具体化。例如，不能在 TE 内执行有关发送一个消息或接收一个超时信号的抽象概念。测试系统的其余部分完成对消息的编码，并分别通过具体的物理手段将之发送出去，或者对时间进行度量，以确定定时器何时到期。

SA 和 PA 及其与 TE 的交互作用在 ITU-T Z.144 建议书[1]中予以定义。TCI 规范用于定义 TE 与 TMC 之间的交互作用。

记录接口为测试系统体系结构中的所有单元提供记录功能，也就是说，TE、TM、CH、CD、SA 和 PA 能够通过 TL 记录有关测试执行情况的信息。图 2 对 TL 做了更加详细的描述。

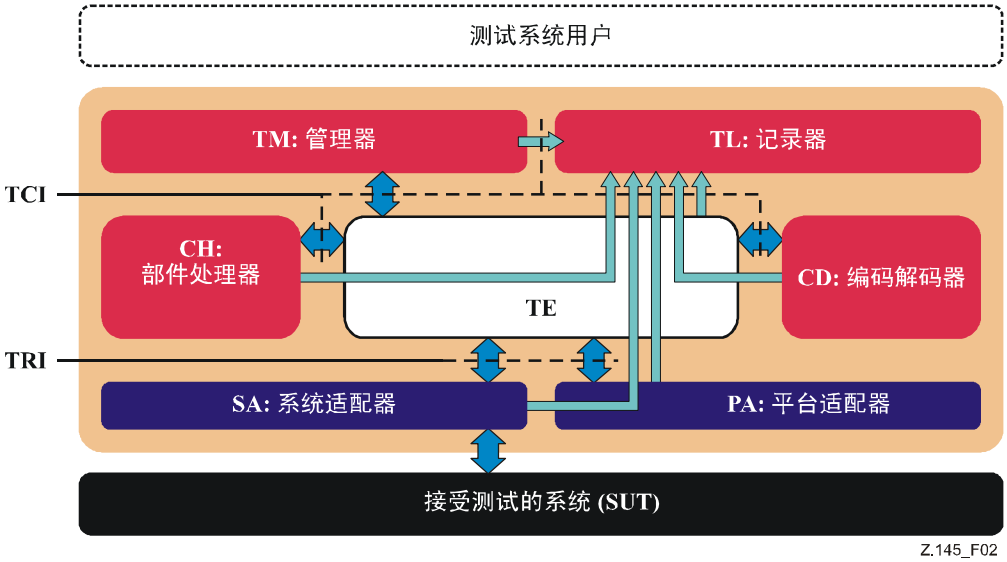

图 2/Z.145－TL详解

6.1.1 测试管理控制器（TMC）

TMC 实体包括与以下管理相关的功能：

- 测试执行；
- 部件；以及
- 编码和解码。

6.1.1.1 测试管理器（TM）

TM 实体负责测试系统总的管理工作。在对测试系统进行初始化后，在 TM 实体内启动执行测试。实体负责正确调用 TTCN-3 模块，也就是说，需要的话，把模块参数传播给 TE，如 IXIT 信息。典型地，该实体还应负责实施一个测试系统用户接口。

6.1.1.2 编码器/解码器（CD）

CD 实体负责把 TTCN-3 值编码和解码为适于发送给接受测试系统的比特串。TE 负责确定使用哪个编码解码器。它把 TTCN-3 数据传送给适当的编码器，以获得编码数据。在 CD 实体中，通过使用适当的解码器，对接收到的数据进行解码，把接收到的数据转换为 TTCN-3 值。

6.1.1.3 部件处理器（CH）

TE 可分布于若干个测试设备上。CH 负责实施各分布式测试系统实体之间的通信。CH 实体提供手段实现各测试系统实体间的同步，这些测试系统实体可能分布于若干个节点上。

注 1 — 节点和测试设备当作同义词使用。

分布于若干节点中的测试系统的一般结构如图 3 所示。

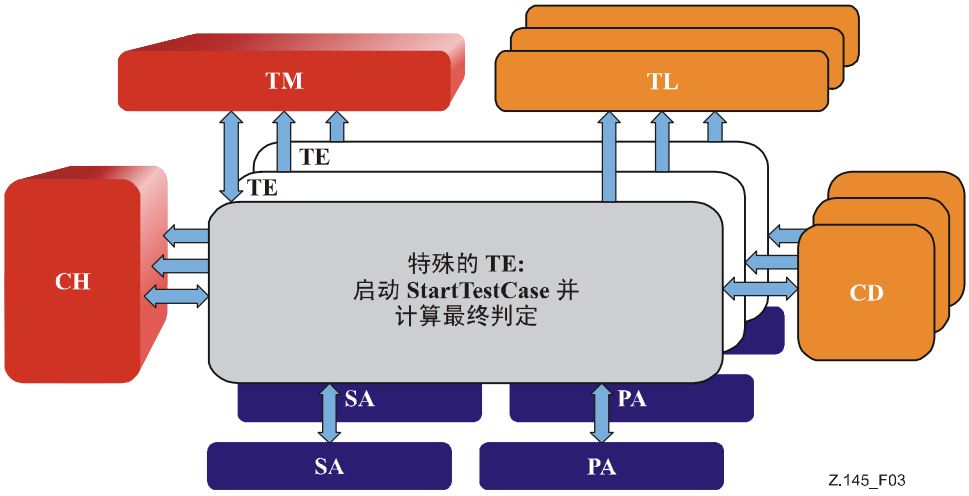

图 3/Z.145－分布式TTCN-3测试系统的一般结构

测试系统内的每个节点包括 TE、SA、PA、CD 和 TL 实体。实体 CH 和 TM 负责解决每个节点上各 TE 之间的测试管理和测试部件处理问题。启动测试用例的 TE 是一个特殊的 TE。它负责计算最终的测试用例判定。除此之外，所有的 TE 以同样的方式进行处理。

注 2 － 在某个特定时间点上，一个测试系统至多只能执行一个测试用例。也就是说，在同一时间，一个TTCN-3模块不能执行多个测试用例。

由 CH 控制创建 TE 中的 MTC、PTC 和控制部件。请注意系统部件的特殊作用，它只在概念上存在，不作为 TE 中的一个运行测试部件。系统端口，即系统部件端口，可以分布于若干个节点上。另外，不同节点上的测试部件可以访问 SUT 的同一物理端口，也就是说，它们可以映射至测试系统接口的同一端口。

例子：经本地代理，TE 可实现对远程实际 SUT 端口的访问。

TTCN-3 部件间的通信可以基于消息，也可以基于过程。因此，CH 会对基于消息和基于过程的 TTCN-3 部件通信进行改造，以便使之适应测试系统的特定执行平台。它知道 TTCN-3 测试部件通信端口间的连接。它应发送请求操作从一个 TTCN-3 部件传播给另一个 TTCN-3 部件。接收部件可以驻留在位于不同节点上的同一 TE 的一个不同实例中。而后通过把各接收到的测试事件排列于 TE 的端口队列中，它把任何接收到的测试事件告知 TE。

TTCN-3 部件间基于过程的通信操作在 CH 上也是可见的。CH 应区分不同类型的、基于过程的通信，即调用、应答和执行，并以适当的方式把它们传播给目标部件驻留其中的 TE。TTCN-3 基于过程的通信语义，即此类操作对 TTCN-3 测试部件执行产生的效果，应在 TE 中予以处理。

为了在若干个节点上分发测试部件，需要额外的通信。部件管理通信包括指明创建测试部件、启动执行测试部件、判定分发，以及指明部件终止。CH 不执行 TTCN-3 部件行为，而是通过 TE 在若干个部件间执行通信。

6.1.1.4 测试记录器（TL）

TL 实体完成测试事件记录，并提交给测试系统用户。它记录有关测试执行情况的信息，如已经创建、启动和终止了哪个测试部件，哪个数据发送给了 SUT，从 SUT 处接收到了哪个数据，哪个数据匹配于 TTCN-3 模板，已经启动、停止或到期哪个定时器等。

6.1.2 可执行TTCN-3（TE）

TE 实体执行或解释一个 TTCN-3 模块。概念上，TE 可被分解为六个相互作用的实体：控制、行为、部件、类型、值和队列实体。TE 的这种结构分解在 ITU-T Z.144 建议书[1]中予以定义。在本建议书中使用在 ITU-T Z.144 建议书[1]中为 TE 定义的术语。

6.1.3 SUT适配器（SA）

SA 是接受测试系统适配器（SA）的实施方案，如[1]中所定义。在本建议书中使用在 ITU-T Z.144 建议书[1]中为 SA 定义的术语。

6.1.4 平台适配器（PA）

PA 是平台适配器（PA）的实施方案，如 ITU-T Z.144 建议书[1]中所定义。在本建议书中使用在 ITU-T Z.144 建议书[1]中为 PA 定义的术语。

6.2 有关TTCN-3测试系统的执行要求

每个 TCI 操作调用都应被认为是调用实体中的一个微小操作。一旦达成预期效果，或者如果未成功完成操作，则执行一个 TCI 操作的被调用实体应把控制归还给调用实体。被调用实体不得妨碍基于过程的通信的实施。

如前所述，对 TTCN-3 测试系统或单个实体是在单个可执行进程中执行还是把它们分布于不同的进程或甚至测试设备中不做任何假设。

一个 TCI 实施方案应满足上述要求。

7 TTCN-3控制接口和操作

本节定义一组抽象数据类型，用于表示 TE 与 TMC 之间进行通信的数据。另外，本节还依据它们的标记定义了 TCI 操作，定义了何时使用它们，以及它们应对 TTCN-3 测试系统产生什么样的效果。

本定义还包括对各个 TCI 操作调用所要求的输入参数及其返回值的更详细说明。

7.1 TCI概述

TCI 定义 TTCN-3 测试系统内可执行 TTCN-3（TE）、部件处理器（CH）、测试管理器（TM）、编码器/解码器（CD）、测试记录器（TL）实体之间的交互作用。它为 TE 提供了完成以下工作的手段：

- 管理测试执行；
- 在不同测试设备间分发测试部件的执行；
- 编码和解码测试数据；以及
- 记录有关测试执行的信息。

TCI 由四个子接口组成：

- **TCI-测试管理器接口（TCI-TM）：**本接口包括管理测试执行、提供模块参数和外部常量以及提供测试事件记录所需的所有操作。
- **TCI-部件处理器接口（TCI-CH）：**本接口由对集中式或分布式测试系统中TTCN-3测试部件实施管理以及完成其间通信所需的各操作组成。它包括用于创建、启动与停止测试部件、用于建立TTCN-3部件间连接、用于管理测试部件及其判定以及用于处理TTCN-3部件间基于消息和基于过程的通信的操作。
- **TCI-编码器/解码器接口（TCI-CD）：**本接口包括获得和访问编码解码器（即编码器或解码器）所需的所有操作，用于对发送数据（使用TTCN-3编码属性进行定义）进行编码，以及用于对接收数据进行解码。
- **TLI-测试记录器接口（TCI-TL）：**本接口包括获得有关测试执行信息以及控制该信息详细程度所需的所有操作。

所有接口都是双向的，这样，调用方和被调用方都可以驻留在测试系统的 TE 和 TMC 中。提供的接口（指的是为 TE 提供接口的那些操作）和要求的接口（指的是需要使用一个来自 TE 的接口的那些操作）结合为有关每个接口各自的、提供的子接口和要求的子接口，即提供的 TCI-TM/要求的 TCI-TM、提供的 TCI-CH/要求的 TCI-CH，提供的 TCI-CD/要求的 TCI-CD，以及提供的 TCI-TL/要求的 TCI-TL。

7.1.1 TTCN-3与TCI操作调用之间的相互关系

对某些 TTCN-3 操作调用，与 TCI 操作调用存在一种直接的相互关系，如表 1 所示。某些 TTCN-3 操作与一对 TCI 操作请求和 TCI 操作相关，以便实现在整个测试系统中传播 TTCN-3 操作。对其它 TCI 操作调用，存在一种间接的相互关系 — 需要它们来实现基本概念的 TTCN-3 语义。

仅当在一个连接于另一个测试部件端口的测试部件端口上调用这些操作时，所示的有关 TTCN-3 通信操作（即发送、调用、应答和引起）的相互关系才存在。在测试部件端口上调用的通信操作的相互关系被映射到测试系统接口端口上，在 ITU-T Z.144 建议书[1]中规定。

表 1/Z.145－TTCN-3与TCI操作调用之间的相互关系

TTCN-3操作名称	TCI操作名称	TCI接口名称
发送	tciSendConnected（参见注 1）	提供的 TCI-CH
	tciSendConnectedBC（参见注 2）	
	tciSendConnectedMC（参见注 3）	
	tciEnqueueMsgConnected	要求的 TCI-CH
调用	tciCallConnected（参见注 1）	提供的 TCI-CH
	tciCallConnectedBC（参见注 2）	
	tciCallConnectedMC（参见注 3）	
	tciEnqueueCallConnected	要求的 TCI-CH
应答	tciReplyConnected（参见注 1）	提供的 TCI-CH
	tciReplyConnectedBC（参见注 2）	
	tciReplyConnectedMC（参见注 3）	
	tciEnqueueReplyConnected	要求的 TCI-CH
引起	tciRaiseConnected（参见注 1）	提供的 TCI-CH
	tciRaiseConnectedBC（参见注 2）	
	tciRaiseConnectedMC（参见注 3）	
	tciEnqueueRaiseConnected	要求的 TCI-CH
创建	tciCreateTestComponentReq	提供的 TCI-CH
	tciCreateTestComponent	要求的 TCI-CH
启动（一个部件）	tciStartTestComponentReq	提供的 TCI-CH
	tciStartTestComponent	要求的 TCI-CH
停止（一个部件）	tciStopTestComponentReq	提供的 TCI-CH
	tciStopTestComponent	要求的 TCI-CH
取消	tciKillTestComponentReq	提供的 TCI-CH
	tciKillTestComponent	要求的 TCI-CH
连接	tciConnectReq	提供的 TCI-CH
	tciConnect	要求的 TCI-CH
断开连接	tciDisconnectReq	提供的 TCI-CH
	tciDisconnect	要求的 TCI-CH
映射	tciMapReq	提供的 TCI-CH
	tciMap	要求的 TCI-CH
取消映射	tciUnmapReq	提供的 TCI-CH
	tciUnmap	要求的 TCI-CH
运行	tciTestComponentRunningReq	提供的 TCI-CH
	tciTestComponentRunning	要求的 TCI-CH

表 1/Z.145－TTCN-3与TCI操作调用之间的相互关系

TTCN-3操作名称	TCI操作名称	TCI接口名称
激活的	tciTestComponentAliveReq	提供的 TCI-CH
	tciTestComponentAlive	要求的 TCI-CH
完成的	tciTestComponentDoneReq	提供的 TCI-CH
	tciTestComponentDone	要求的 TCI-CH
取消的	tciTestComponentKilledReq	提供的 TCI-CH
	tciTestComponentKilled	要求的 TCI-CH
mtc	tciGetMTCReq	提供的 TCI-CH
	tciGetMTC	要求的 TCI-CH
执行	tciTestCaseExecuteReq	提供的 TCI-CH
	tciTestCaseExecute	要求的 TCI-CH
记录	tliLog	提供的 TCI-TL
注 1 － 用于单播通信。 注 2 － 用于广播通信。 注 3 － 用于多播通信。		

7.2 TCI数据

TCI 规范定义一组抽象数据类型。它们在很高层次上描述哪种数据应从一个调用实体传递给一个被调用实体。抽象数据类型用于确定：

- TTCN-3数据如何从一个TE传递给一个编码器，以便把TTCN-3值表示编码为一个比特串，反之亦然；
- 从一个解码器传递给TE的数据如何从一个比特串解码为其TTCN-3值表示。

为这些抽象数据类型定义了一组操作，以便编码器/解码器对数据进行处理。

这些抽象数据类型的具体表示以及基本数据类型的定义，如 `string` 和 `boolean` 在第 8 节和第 9 节各自的语言映射中进行定义。

注意：在测试系统实施方案中，任何标识符数据类型的值都应是唯一的，唯一性定义为在任何时间点上都是全局不同的。这确保由不同的标识符来标识不同的对象，如两个定时器，以及不得重用标识符。

7.2.1 通用抽象数据类型

定义以下抽象数据类型，用于定义 TCI 操作。

7.2.1.1 管理

`TciModuleIdType` `TciModuleIdType` 的值为 TTCN-3 模块的名称，如 TTCN-3 ATS 中规定。该抽象类型用于模块处理。

`TciModuleParameterIdType` `TciModuleParameterIdType` 的值为 TTCN-3 模块参数的合格名称，如 TTCN-3 ATS 中规定。该抽象类型用于模块参数处理。

`TciTestCaseIdType` `TciTestCaseIdType` 的值为 TTCN-3 测试用例的合格名称，如 TTCN-3 ATS 中规定。该抽象类型用于测试用例处理。

`TciModuleIdListType` `TciModuleIdListType` 类型的值为一个 `TciModuleIdType` 清单。当获取由 TTCN-3 模块输入的模块清单时，使用该抽象类型。

`TciModuleParameterType` `TciModuleParameterType` 类型的值为一个 `TciModuleParameterIdType` 和 `Value` 结构。该抽象类型用于表示参数名称以及模块参数的缺省值。

TciModuleParameterListType	TciModuleParameterListType 类型的值为一个 TciModuleParameterType 清单。当获取 TTCN-3 模块的模块参数时，使用该抽象类型。
TciParameterType	TciParameterType 类型的值包括一个 TTCN-3 Value 以及一个 TciParameterPassingModeType 值，用于表示为 TTCN-3 ATS 中参数规定的参数传递模式。
TciParameterPassingModeType	TciParameterPassingModeType 类型的值为 IN、INOUT 或 OUT。当指明启动一个测试用例时或当指明终止一个测试用例时，使用该抽象类型。
TciParameterListType	TciParameterListType 类型的值为一个 TciParameterType 清单。当指明启动一个测试用例时或当指明终止一个测试用例时，使用该抽象类型。
TciParameterTypeType	TciParameterTypeType 类型的值为一个 Type 和 TciParameterPassingModeType 结构。该抽象类型用于表示类型以及测试用例参数的参数传递模式。
TciParameterTypeListType	TciParameterTypeListType 类型的值为一个 TciParameterTypeType 清单。该抽象类型用于表示测试用例的参数清单。
TciTestComponentKindType	TciTestComponentKindType 类型的值为 TTCN-3 测试部件类型组的文字表示，即 MTC、PTC、PTC_ALIVE 和 CONTROL。该抽象类型用于部件处理。
TciTypeClassType	TciTypeClassType 类型的值为 TTCN-3 中类型类别组的文字表示，如 boolean、float、record 等。该抽象类型用于值处理。

7.2.1.2 通信

TciBehaviourIdType	用于确定某个TTCN-3行为功能的TciBehaviourIdType类型值。

带前缀 Tri 的其它抽象数据类型来自 ITU-T Z.144 建议书[1]：TriPortIdType、TriPortIdListType、TriComponentIdType、TriComponentIdListType、TriAddressType、TriAddressListType 和 TriMessageType。

7.2.2 抽象TTCN-3数据类型和值

本节定义抽象数据类型组，以创建 TTCN-3 类型和值表示。通过一组附随的操作来定义各数据类型的功能。对该抽象数据类型的操作或使用该抽象数据类型的操作返回一个该抽象类型值，或者返回一个基本类型，如 boolean。

已用接口描述语言（IDL）定义了所有操作。在第 8 节和第 9 节中给出有关抽象数据类型操作的、具体的语言映射。在某些语言中，有关抽象数据类型操作的应用通过把具体的值作为参数传递（通过值或通过引用来传递，这取决于映射关系）给操作来表示。其它的语言可能对具体的值选择其它引用方法，例如，通过把值当作一个对象，对该对象引入一个对应操作的方法。为了指明不能执行某个任务或者指明之后不存在某个可选的参数，使用特别的值 null。可将之当作一个保留值，用于指明一个特殊的值。语言映射应定义该特别值 null 的具体表示。

抽象 TTCN-3 类型和值表示由两部分组成：

- 一个用于表示TTCN-3模块中所有TTCN-3类型的抽象数据类型Type；
- 表示TTCN-3值的不同抽象数据类型，即某个特定TTCN-3类型的TTCN-3值。这可以是TTCN-3预定义类型的值，或者是TTCN-3用户定义类型的值。

为了访问、评估和编码 TTCN-3 数据，测试系统使用抽象数据类型 Type 和不同的抽象值数据类型。因此，这些抽象数据类型定义可执行 TTCN-3（TE）与其余使用 TCI 接口的测试系统之间的抽象程度。

7.2.2.1 抽象TTCN-3数据类型

依据本建议书，TTCN-3 类型，预定义的或是用户定义的，应在 TCI 接口上使用抽象数据类型 Type 来表示。

为抽象数据类型定义了一组操作，用以：

- 引用预定义的和用户定义的TTCN-3数据类型；以及
- 创建和保留TTCN-3值。

为抽象数据类型定义了以下操作：

`TciModuleIdType getDefiningModule()`	返回模块的模块标识符，在该模块中定义类型。如果类型是一个 TTCN-3 基本类型，如布尔型、整数型等，则返回特别值 `null`。
`TString getName()`	返回类型的名称，如 TTCN-3 模块中所定义。
`TciTypeClassType getTypeClass()`	返回各自类型的类型类别。`TciTypeClassType` 的值可以是以下常量之一：`ADDRESS`、`ANYTYPE`、`BITSTRING`、`BOOLEAN`、`CHARSTRING`、`COMPONENT`、`ENUMERATED`、`FLOAT`、`HEXSTRING`、`INTEGER`、`OBJID`、`OCTETSTRING`、`RECORD`、`RECORD_OF`、`SET`、`SET_OF`、`UNION`、`UNIVERSAL_CHARSTRING`、`VERDICT`。
`Value newInstance()`	返回给定类型的新建值。未定义创建值的该初始值。
`TString getTypeEncoding()`	返回类型编码属性，如 TTCN-3 模块中所定义。
`TString getTypeEncodingVariant()`	如果存在的话，该操作返回值编码变量属性，如 TTCN-3 中所定义。如果未定义任何编码变量属性，则返回特别值 `null`。
`TStringseq getTypeExtension()`	返回类型扩展属性，如 TTCN-3 模块中所定义。

7.2.2.2 抽象TTCN-3值

依据本建议书，通过众多抽象数据类型，在 TCI 接口处来描述 TTCN-3 值。

图 4 描述了有关 TTCN-3 值（短：抽象值）的抽象数据类型之间的层次结构。

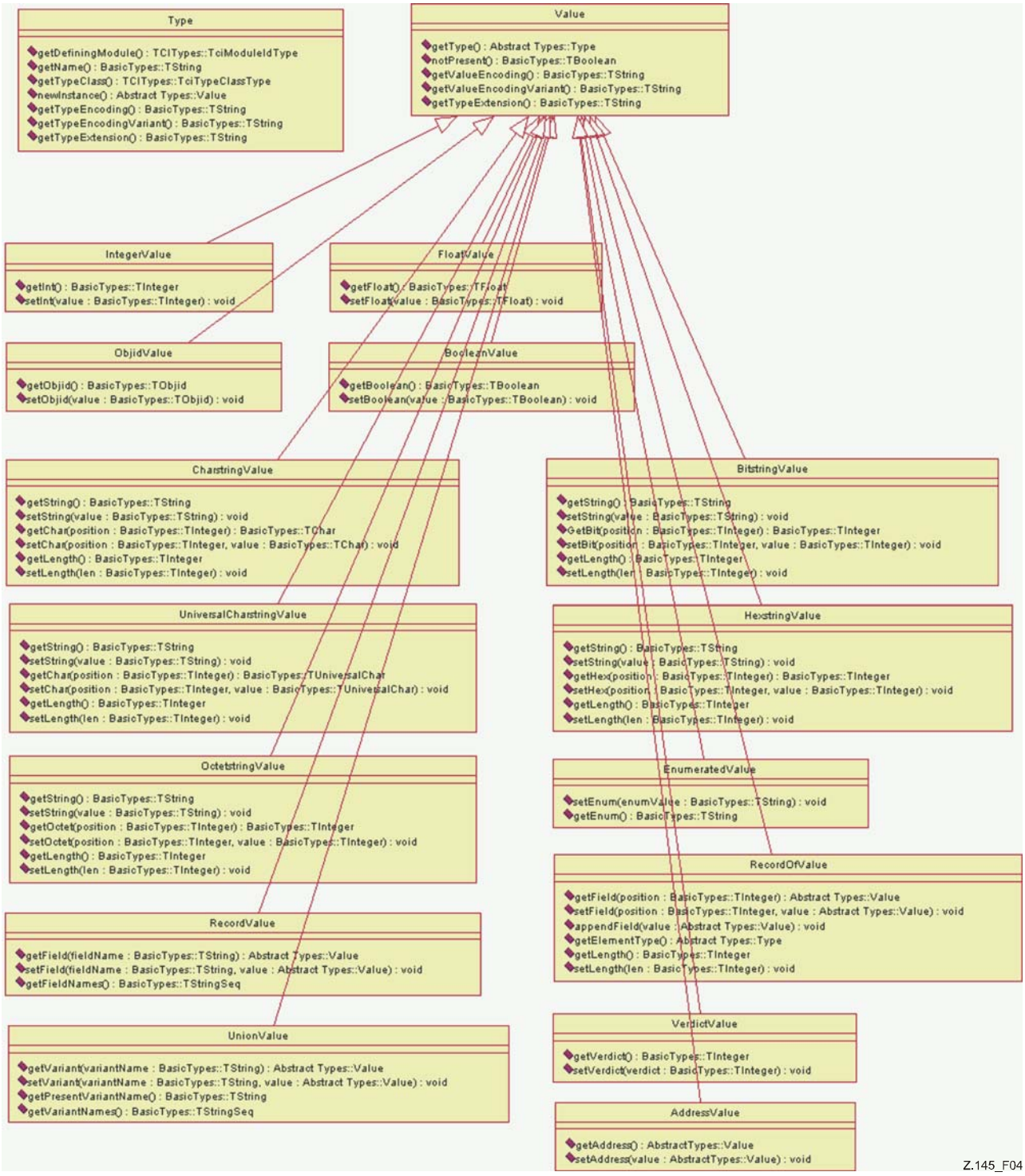

图 4/Z.145－抽象值的层次结构

如图 4 所示，所有的 TTCN-3 抽象值共享同一基本抽象数据类型 `Value`。在本公共基本数据类型上，也为源自它的抽象值类型定义了所有操作。

7.2.2.2.1 抽象数据类型`Value`

在基本抽象数据类型 `Value` 上定义了以下操作。在各自的语言映射部分，对这些操作的具体表示做了定义。

`Type getType()` 返回规定值的类型。

`TBoolean notPresent()` 如果规定值为 `omit`，则返回 `true`，否则返回 `false`。

`TString getValueEncoding()`	如果存在的话，返回值编码属性，如 TTCN-3 中所定义。如果未定义任何编码属性，则返回特别值 `null`。
`TString getValueEncodingVariant()`	如果存在的话，返回值编码变量属性，如 TTCN-3 中所定义。如果未定义任何编码变量属性，则返回特别值 `null`。

7.2.2.2.2 抽象数据类型IntegerValue

抽象数据类型 `IntegerValue` 基于抽象数据类型 `Value`。它表示 TTCN-3 整数值。

在抽象数据类型 `IntegerValue` 上定义了以下操作：

`TInteger getInt()`	返回该 TTCN-3 整数的整数值。
`void setInt(in TInteger value)`	把该 `IntegerValue` 置为 `value`。

7.2.2.2.3 抽象数据类型FloatValue

抽象数据类型 `FloatValue` 基于抽象数据类型 `Value`。它表示 TTCN-3 浮点数值。

在抽象数据类型 `FloatValue` 上定义了以下操作：

`TFloat getFloat()`	返回该 TTCN-3 浮点数的浮点数值。
`void setFloat(in TFloat value)`	把该 `FloatValue` 置为 `value`。

7.2.2.2.4 抽象数据类型BooleanValue

抽象数据类型 `BooleanValue` 基于抽象数据类型 `Value`。它表示 TTCN-3 布尔值。

在抽象数据类型 `BooleanValue` 上定义了以下操作：

`TBoolean getBoolean()`	返回该 TTCN-3 布尔数的布尔值。
`void setBoolean(in TBoolean value)`	把该布尔值置为 `value`。

7.2.2.2.5 抽象数据类型ObjidValue

抽象数据类型 `ObjidValue` 基于抽象数据类型 `Value`。它表示 TTCN-3 对象标识符值。

在抽象数据类型 `ObjidValue` 上定义了以下操作：

`TObjid getObjid()`	返回该 TTCN-3 对象标识符的对象标识符值。
`void setObjid(in TObjid value)`	把该 `ObjidValue` 置为 `value`。

7.2.2.2.6 抽象数据类型CharstringValue

抽象数据类型 `CharstringValue` 基于抽象数据类型 `Value`。它表示 TTCN-3 字符串值。`Tchar` 为字符串值中的一个字符。

在抽象数据类型 `CharstringValue` 上定义了以下操作：

`TString getString()`	返回 TTCN-3 `charstring` 的字符串值。空 TTCN-3 `charstring` 的文本表示为`''`，长度为 0。
`void setString(in TString value)`	把该 `CharstringValue` 置为 `value`。
`TChar getChar(in TInteger position)`	返回 `position` 处 TTCN-3 `charstring` 的字符值。位置 `0` 表示 TTCN-3 `charstring` 的第一个字符。位置的有效值为 `0` 到长度－`1`。
`void setChar(in TInteger position, in TChar value)`	设置 position 处的字符值。位置的有效值为 `0` 到长度－`1`。
`TInteger getLength()`	返回该 CharstringValue 的长度，以字符数计，如果该 CharstringValue 值为 omit，则长度为 0。
`void setLength(in TInteger len)`	setLength 首先把该 CharstringValue 重置为其初始值，之后把该 `CharstringValue` 的长度设置 `len`，以字符数计。

7.2.2.2.7 抽象数据类型UniversalCharstringValue

抽象数据类型 UniversalCharstringValue 基于抽象数据类型 Value。它表示 TTCN-3 通用字符串值。TUniversalChar 为通用字符串值中的一个字符。

在抽象数据类型 UniversalCharstringValue 上定义了以下操作：

TString getString()	返回该 UniversalCharstringValue 的文本表示，如 TTCN-3 中所定义。
void setString(in TString value)	依据 value 所定义的文本表示，设置该 UniversalCharstringValue 的值。
TUniversalChar getChar(in TInteger position)	返回 position 处 TTCN-3 universal charstring 的通用字符值。位置 0 表示 TTCN-3 universal charstring 的第一个 TUniversalChar。位置的有效值为 0 到长度－1。
void setChar(in TInteger position, in TUniversalChar value)	设置 position 处通用字符的 value。位置的有效值为 0 到长度－1。
TInteger getLength()	返回该通用字符串值的长度，以通用字符计，如果该通用字符串值为 omit，则长度为 0。
void setLength(in TInteger len)	setLength 首先把该 UniversalCharstringValue 重置为其初始值，之后把该 UniversalCharstringValue 的长度置为 len，以通用字符计。

7.2.2.2.8 抽象数据类型BitstringValue

抽象数据类型 BitstringValue 基于抽象数据类型 Value。它表示 TTCN-3 比特串值。

在抽象数据类型 BitstringValue 上定义了以下操作：

TString getString()	返回该 BitstringValue 的文本表示，如 TTCN-3 中所定义。例如，0101 的文本表示为“0101”B。空 TTCN-3 bitstring 的文本表示为""B，其长度为 0。
void setString(in TString value)	依据 value 所定义的文本表示，设置该 BitstringValue 的值。例如，如果 value 的文本表示为“0101”B，则该 BitstringValue 的值为 0101。
TChar getBit(in TInteger position)	返回该 TTCN-3 比特串 position 处的值（0|1），作为一个字符。位置 0 表示 TTCN-3 比特串的第一位。位置的有效值为 0 到长度－1。
void setBit(in TInteger position, TInteger value)	把 position 处的位置为值（0|1）。位置 0 表示该 BitstringValue 的第一位。位置的有效值为 0 到长度－1。
TInteger getLength()	返回该 BitstringValue 的长度，以比特数计。如果该 BitstringValue 值为 omit，则长度为 0。
void setLength(in TInteger len)	setLength 首先把该 BitstringValue 重置为其初始值，之后把该 BitstringValue 的长度置为 len，以比特数计。

7.2.2.2.9 抽象数据类型OctetstringValue

抽象数据类型 OctetstringValue 基于抽象数据类型 Value。它表示 TTCN-3 八位字节串值。

在抽象数据类型 OctetstringValue 上定义了以下操作：

TString getString()	返回该 OctetstringValue 的文本表示，如 TTCN-3 中所定义。例如，0xCAFFEE 的文本表示为“CAFFEE”O。空 TTCN-3 八位字节串的文本表示为""O，其长度为 0。
void setString(in TString value)	依据 value 定义的文本表示，设置该 OctetstringValue 的值。例如，如果 value 中的文本表示为“CAFFEE”O，则该 OctetstringValue 的值为 0xCAFFEE。
TChar getOctet(in TInteger position)	返回该 TTCN-3 八位字节串 position 处的值（0..255）。位置 0 表示 TTCN-3 八位字节串的第一个八位字节。位置的有效值为 0 到长度－1。
void setOctet(in TInteger position, in TInteger value)	把 position 处的八位字节置为值（0..255）。位置 0 表示八位字节串中的第一个八位字节。位置的有效值为 0 到长度－1。
TInteger getLength()	返回该 OctetstringValue 的长度，以八位字节数计。如果该 OctetstringValue 值为 omit，则长度为 0。
void setLength(in TInteger len)	setLength 首先把该 OctetstringValue 重置为其初始值，之后把该 OctetstringValue 的长度置为 len，以八位字节数计。

7.2.2.2.10 抽象数据类型HexstringValue

抽象数据类型 HexstringValue 基于抽象数据类型 Value。它表示 TTCN-3 十六进制串值。

在抽象数据类型 HexstringValue 上规定了以下操作：

TString getString()	返回该 HexstringValue 的文本表示，如 TTCN-3 中规定。例如，0xAFFEE 的文本表示为“AFFEE”H。空 TTCN-3 十六进制串的文本表示为""H，其长度为 0。
void setString(in TString value)	依据 value 规定的文本表示，设置该 HexstringValue 的值。例如，如果 value 中的文本表示为“AFFEE”H，则该 HexstringValue 的值为 0xAFFEE。
TChar getHex(in TInteger position)	返回该 TTCN-3 十六进制串 position 处的值（0..15）。位置 0 表示 TTCN-3 十六进制串的第一个十六进制数字。位置的有效值为 0 到长度－1。
void setHex(in TInteger position, in TInteger value)	把 position 处的十六进制数字置为值（0..15）。位置 0 表示十六进制串中的第一个八位字节。位置的有效值为 0 到长度－1。
TInteger getLength()	返回该 HexstringValue 的长度，以八位字节数计。如果该 HexstringValue 值为 omit，则长度为 0。
void setLength(in TInteger len)	setLength 首先把该 HexstringValue 重置为其初始值，之后把该 HexstringValue 的长度置为 len，以十六进制数字数计。

7.2.2.2.11 抽象数据类型RecordValue

抽象数据类型 RecordValue 基于抽象数据类型 Value。它规范如何获取和设置 TTCN-3 记录类型。同一抽象数据类型适用于类型类别为 SET 的值。record 与 set 之间的差别只在匹配时有关。

在抽象数据类型 RecordValue 上规定了以下操作：

Value getField(in TString fieldName)	返回字段名称 fieldName 的值。由于记录字段可以拥有 TTCN-3 中规定的任何类型，因此返回值为公共抽象基本类型 Value。如果没有从记录中获得字段，则返回特别值 null。

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

把 record 字段名称 `fieldName` 置为 `value`。对一个字段如何保存于一个记录中不得做任何假设。可以选择内部执行来保存对该值的引用或者拷贝该值。假设拷贝值是安全的。因此，应假设对 `value` 的后续修改不会反映在 record 中。

`TStringSeq getFieldNames()` 返回一个有关字段名的字符串序列，如果记录没有字段，则返回空序列。

7.2.2.2.12 抽象数据类型RecordOfValue

抽象数据类型 `RecordOfValue` 基于抽象数据类型 `Value`。它规范如何获取和设置 TTCN-3 记录类型。同一抽象数据类型适用于类型类别为 `SET_OF` 的值。`record of` 与 `set of` 之间的差别只在匹配时有关。

在抽象数据类型 `RecordOfValue` 上规定了以下操作：

`Value getField(in TInteger position)` 如果位置处于 `0` 与长度－`1` 之间，则返回 `position` 处的 `record of` 值，否则返回特别值 `null`。由于 `record of` 可以拥有 TTCN-3 中规定任何类型的字段，因此返回值为公共抽象基本类型 `Value`。

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

把 `position` 处的字段置为 `value`。如果位置大于（长度－`1`），则 `record of` 的长度扩展为（`position`＋`1`）。忽略 `length` 和 `position`－`1` 上初始位置间的 `record of` 单元。对一个字段如何保存于一个 `record of` 中不得做任何假设。可以选择内部执行来保存对该值的引用或者拷贝该值。假设拷贝值是安全的。因此，应假设对 `value` 的后续修改不会反映在 `record of` 中。

`void appendField(in Value value)` 在 `record of` 末尾添加 value，即在位置 `length` 处。对一个字段如何保存于一个 `record of` 中不得做任何假设。可以选择内部执行来保存对该值的引用或者拷贝该值。假设拷贝值是安全的。因此，应假设对 `value` 的后续修改不会反映在 `record of` 中。

`Type getElementType()` 返回该 `record of` 的单元类型。

`TInteger getLength()` 返回 `record of` value 的实际长度，如果 `record of` value 为 omit，则长度为 0。

`void setLength(in TInteger len)` 把 `record of` 长度置为 `len`。如果 `len` 大于初始长度，则新建单元值为 `omit`。如果 `len` 小于或等于初始长度，则忽略该操作。

7.2.2.2.13 抽象数据类型UnionValue

抽象数据类型 `UnionValue` 基于抽象数据类型 `Value`。它规范如何获取和设置 TTCN-3 联合类型中的变量。TTCN-3 `anytype` 通过 `UnionValue` 来表示，其中，通过 `getType()` 获得的类型的类型类别为 `ANYTYPE`。有关类型类别的详细内容，参见第 7.2.2.1 节。

在抽象数据类型 `UnionValue` 上规定了以下操作：

`Value getVariant(in TString variantName)`

如果 `variantName` 等于 `getPresentVariantName` 的结果，则返回 TTCN-3 联合 `variantName` 的值，否则返回特别值 `null`。`variantName` 表示 TTCN-3 中规定的联合变量的名称。

`void setVariant(in TString variantName, in Value value)`	把联合的 variantName 置为 value。如果对该联合未规定 variantName，则忽略该操作。如果选择了另一个变量，则改为选择新的变量。
`TString getPresentVariantName()`	返回一个表示给定 TTCN-3 联合中当前所选变量名的字符串。如果未选择任何变量，则返回特别值 null。
`TStringSeq getVariantNames()`	返回一个变量名字符串序列，如果该联合没有任何字段，则返回特别值 null。如果 UnionValue 表示 TTCN-3 的任何类型，即通过 getType()获得的类型的类型类别为 ANYTYPE，则返回所有预定义的和用户规定的 TTCN-3 类型。

7.2.2.2.14 抽象数据类型EnumeratedValue

抽象数据类型 EnumeratedValue 基于抽象数据类型 Value。它规范如何设置和获取枚举的 TTCN-3。

在抽象数据类型 EnumeratedValue 上规定了以下操作：

`TString getEnum()`	返回该 EnumeratedValue 的字符串标识符。该标识符等同 TTCN-3 规范中的标识符。
`void setEnum(in TString enumValue)`	把 enum 置为 enumValue。如果对该枚举，enumValue 不是一个允许的值，则忽略操作。

7.2.2.2.15 抽象数据类型VerdictValue

抽象数据类型 VerdictValue 基于抽象数据类型 Value。它规范如何设置和获取 TTCN-3 判定。

在抽象数据类型 VerdictValue 上规定了以下操作：

`TInteger getVerdict()`	返回该 VerdictValue 的整数值。整数为以下常量之一：ERROR、FAIL、INCONC、NONE、PASS。
`void setVerdict(in TInteger verdict)`	把该 VerdictValue 置为判定。注意：任何时候，VerdictValue 都可置为任何上面所述的判定。VerdictValue 不能执行 TTCN-3 中规定的任何判定计算。例如，首先把 VerdictValue 置为 ERROR 而后置为 PASS 是合法的。

7.2.2.2.16 抽象数据类型AddressValue

在基本抽象数据类型 AddressValue 上规定了以下操作。在各自的语言映射部分，对这些操作的具体表示做了规定：

`Value getAddress()`	返回地址值，它应不再是类型类别 ADDRESS，而是用于地址的实际类型。
`void setAddress(in Value value)`	把该地址值置为 value。

7.2.3 抽象记录类型

7.2.3.1 抽象数据类型TciValueTemplate

在抽象数据类型 TciValueTemplate 上规定了以下操作。在各自的语言映射部分，对这些操作的具体表示做了规定：

`boolean isOmit()`	如果模板为一个 omit 模板，则返回 true。
`boolean isAny()`	如果模板为一个 any 模板，则返回 true。
`boolean isAnyOrOmit()`	如果模板为一个 any 或 omit 模板，则返回 true。
`TString getTemplateDef()`	返回该模板的定义。

7.2.3.2 抽象数据类型TciNonValueTemplate

在抽象数据类型 TciNonValueTemplate 上规定了以下操作。在各自的语言映射部分，对这些操作的具体表示做了规定：

boolean isAny()	如果模板为一个 any 模板，则返回 true。
boolean isAll()	如果模板为一个 all 模板，则返回 true。
TString getTemplateDef()	返回该模板的定义。

7.2.3.3 值列表和失配类型

规定了以下抽象数据类型，用于记录值与模板之间的差异。

TciValueList	TciValueList 的一个值为一个值的清单。
TciValueDifference	TciValueDifference 的一个值为一个结构，它包含一个值、一个模板和一个有关该差异理由的描述。
TciValueDifferenceList	TciValueDifferenceList 的一个值为一个值差异的序列。

7.3 TCI操作

本节规定可执行TTCN-3把提供给测试系统的操作(*要求的操作*)以及测试系统把提供给可执行TTCN-3的功能（*提供的操作*）。

术语“要求的”和“提供的”反映了以下事实，即本建议书从用户观点出发规定了对可执行 TTCN-3 的要求。用户对可执行 TTCN-3 提出了某些功能“要求”，以便建立一个完整的、基于 TTCN-3 的测试系统。为了完成其任务，可执行 TTCN-3 须把某些事件告知用户，当中用户须把这种可能性“提供”给可执行 TTCN-3。

本节中的所有操作定义使用接口定义语言（IDL）来规定。具体的语言映射关系在第 8 节和第 9 节中进行规定。附件 B 为记录接口提供了一种可选的、与 XML 的映射关系。

对每一个 TCI 操作调用，列于特定操作定义中的所有 *in*、*inout* 和 *out* 参数都是强制性的。*in* 参数的值由调用实体来规定。调用实体指的是调用的方向。对*要求的*接口上的操作，调用实体为测试系统，而被调用实体为可执行 TTCN-3。对*提供的*接口上的操作，调用实体为可执行 TTCN-3，而测试系统为被调用实体。

同样，*out* 参数的值由被调用实体来规定。在 *inout* 参数情况下，值首先由调用实体来规定，但可以由被调用实体替换为一个新的值。注意：尽管 TTCN-3 也对标记定义使用 *in*、*inout* 和 *out*，但在 TCI IDL 规范中使用的标记与在 TTCN-3 规范中使用的标记是无关的。

操作调用应使用一个保留值来指明不存在参数。有关这些类型的保留值在各个语言映射中予以规定，之后指的应是 null 值。

另外，null 值也应用于指明不能执行某个任务。

由于本节只规定接口而不建议有关如何执行规定功能的具体实施方案，因此术语“实体”应用于确定实现该接口并执行所请求功能的那部分测试系统实施方案。例如，tciSendConnected 操作中的调用实体为 TE，即提供 TE 功能的那部分测试系统实施方案。

接口中的所有功能使用以下模板来描述。删去了不适用于某些操作的描述。

标记	IDL Signature
输入参数	对从调用实体传递给被调用实体的数据的描述，所传数据作为操作的参数。
输出参数	对从被调用实体传递给调用实体的数据的描述，所传数据作为操作的参数。
输入输出参数	对从调用实体传递给被调用实体以及从被调用实体传递给调用实体的数据的描述，所传数据作为操作的参数。
返回值	对从操作返回给调用实体的数据的描述。
约束条件	当操作被调用时，对任何约束条件的描述。
效果	在操作可返回前，对被调用实体要求的行为。

7.3.1 TCI-TM接口

TCI 测试管理接口（TCI-TM）描述要求可执行 TTCN-3 执行的操作，以及测试管理实施方案应提供给 TE 的操作（图 5）。

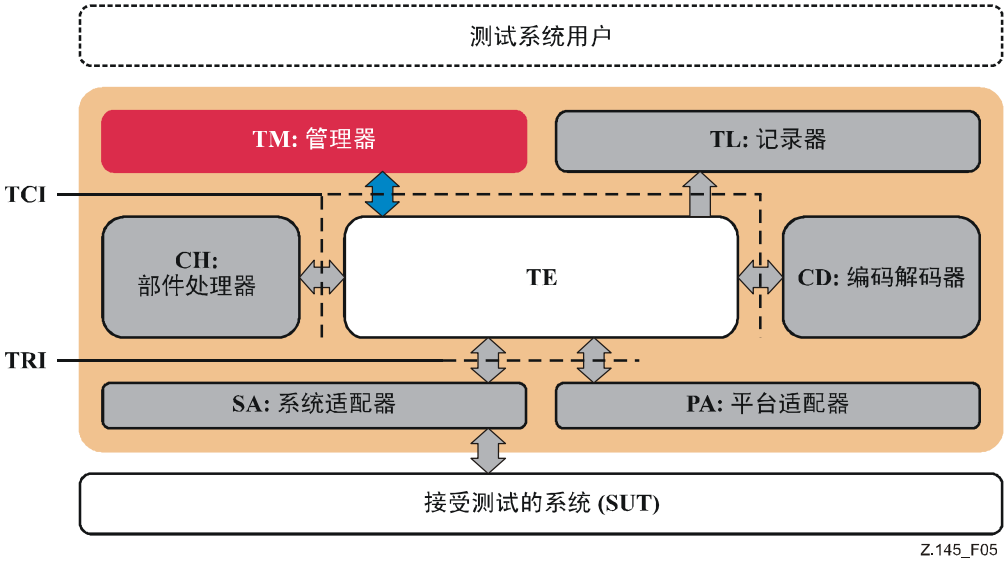

图 5/Z.145－TCI-TM接口

测试管理器实施方案提供对测试系统用户总的测试管理。它要求 TE 提供操作，以启动和停止测试执行某个 TTCN-3 模块或 TTCN-3 模块中的某个测试用例。反过来，它为 TE 提供操作，以解析运行时的模块参数并指明执行终止。

第 10 节描述 TE 或测试管理器操作调用的用法和顺序。

7.3.1.1 要求的TCI-TM

本节规定 TM 对 TE 要求的各操作。除了本节中规定的各操作，测试管理器还要求对 TCI-CD 接口所要求的各操作。

7.3.1.1.1 tciRootModule

标记	void `tciRootModule(in TciModuleIdType moduleName)`	
输入参数	`moduleName`	表示模块标识符的 `moduleName`，如 TTCN-3 中规定。
返回值	空	
约束条件	只有当既没有控制部分正在执行也没有测试用例正在执行时才能使用。	
效果	使用 `tciStartTestCase` 或 `tciStartControl`，通过一个后续调用，`tciRootModule` 选择执行指明的模块。如果不存在任何此类模块，则应发布一个 `tciError`。	

7.3.1.1.2 getImportedModules

标记	`TciModuleIdListType getImportedModules()`
返回值	一个有关根模块所有输入模块的清单。当出现在 TTCN-3 模块中时对各模块进行排序。如果不存在任何输入模块，则返回一个空的模块清单。
约束条件	只有当之前已设置了一个根模块时才能使用。
效果	TE 向管理器提供一个有关根模块输入模块的清单。如果不存在任何输入模块，则返回一个空的模块清单。如果 TE 不能提供一个清单，则应返回特别值 `null`。

7.3.1.1.3 tciGetModuleParameters

标记	`TciModuleParameterListType tciGetModuleParameters(in TciModuleIdType moduleName)`	
输入参数	`moduleName`	`moduleName` 表示模块标识符，应获取该模块的模块参数。
返回值	一个有关标识模块所有模块参数的清单。当出现在 TTCN-3 模块中时对各参数进行排序。如果不存在任何参数，则返回一个空的模块参数清单。	
约束条件	只有当之前已设置了一个根模块时才能使用。	
效果	TE 向管理器提供一个有关确定模块之模块参数的清单。如果不存在任何模块参数，则返回一个空的模块参数清单。如果 TE 不能提供一个清单，则应返回特别值 `null`。	

7.3.1.1.4 tciGetTestCases

标记	`TciTestCaseIdListType tciGetTestCases()`
返回值	一个有关所有测试用例的清单，它们在根模块中规定或输入根模块中。
约束条件	只有当之前已设置了一个根模块时才能使用。
效果	TE 向管理器提供一个有关测试用例的清单。如果不存在任何测试用例，则返回一个空的测试用例清单。如果 TE 不能提供一个清单，则应返回特别值 `null`。

7.3.1.1.5 tciGetTestCaseParameters

标记	`TciParameterTypeListType tciGetTestCaseParameters(in TciTestCaseIdType testCaseId)`	
输入参数	`testCaseId`	一个测试用例标识符，如 TTCN-3 模块中规定。
返回值	一个有关特定测试用例所有参数类型的清单。当出现在测试用例的 TTCN-3 标记中时对各参数类型进行排序。如果不存在任何参数，则返回一个空的参数类型清单。	
约束条件	只有当之前已设置了一个根模块时才能使用。	
效果	TE 向管理器提供一个有关给定测试用例之参数类型的清单。如果不存在任何测试用例参数，则返回一个空的参数类型清单。如果 TE 不能提供一个清单，则应返回特别值 `null`。	

7.3.1.1.6 tciGetTestCaseTSI

标记	`TriPortIdListType tciGetTestCaseTSI(in TciTestCaseIdType testCaseId)`	
输入参数	`testCaseId`	一个测试用例标识符，如 TTCN-3 模块中规定。
返回值	一个有关给定测试用例之所有系统端口的清单，它们已在有关测试用例的系统部件定义中予以声明，即 TSI 端口。如果未为测试用例显性规定系统部件，则清单包含 MTC 测试部件的所有通信端口。当出现在各自的 TTCN-3 部件类型声明中时对各端口进行排序。如果不存在任何系统端口，则返回一个空的清单，即返回一个长度为 0 的清单。	
约束条件	只有当之前已设置了一个根模块时才能使用。	
效果	TE 向管理器提供一个有关给定测试用例之系统端口的清单。如果不存在任何系统端口，则返回一个空的端口清单。如果 TE 不能提供一个清单，则应返回特别值 `null`。	

7.3.1.1.7 tciStartTestCase

标记	`void tciStartTestCase(in TciTestCaseIdType testCaseId, in TciParameterListType parameterList)`	
输入参数	`testCaseId`	一个测试用例标识符，如 TTCN-3 模块中规定。
	`parameterList`	一个有关值的清单，当中的每个值规定一个来自参数清单的参数，如 TTCN-3 测试用例定义中规定。当出现在测试用例的 TTCN-3 标记中时对参数清单中的各参数进行排序。如果没有任何参数需要传递，则传递 `null` 值或一个空的参数清单，即传递一个长度为 0 的清单。
返回值	空	
约束条件	只有当之前已经选择了一个模块时才能被调用。只传递已在当前选择之 TTCN-3 模块中做了声明的测试用例的 `testCaseId`。不能启动输入所引用模块中的测试用例。为了启动输入的测试用例，必须首先使用 `tciRootModule` 操作选择所引用（输入的）模块。	
效果	`tciStartTestCase` 使用给定的参数在当前选择的模块中启动一个测试用例。如果不存在任何此类测试用例，则应发布一个 `tciError`。 参数清单中的所有 *in* 和 *inout* 测试用例参数都包含 `Value`。参数清单中的所有 *out* 测试用例参数都包含特别值 `null`，原因是，只有当测试用例终止时，它们才相关。	

7.3.1.1.8 tciStopTestCase

标记	`void tciStopTestCase()`
返回值	空
约束条件	只有当之前已经选择了一个模块时才能被调用。
效果	`tciStopTestCase` 停止当前正在执行的测试用例。如果 TE 未在执行一个测试用例，则操作应被忽略。如果控制部分正在执行，则 `tciStopTestCase` 应停止执行当前正在执行的测试用例，即最近已使用提供的操作 `tciTestCaseStarted` 指明正在执行测试用例。一个可能正在执行的控制部分应继续执行，好像测试用例已被正常停止，并返回判定 `ERROR`。

7.3.1.1.9 tciStartControl

标记	`TriComponentId tciStartControl()`
返回值	一个 `TriComponentId`，用于表示测试部件。执行模块控制部分。如果 TE 不能启动所选模块的控制部分，则应返回特别值 `null`。
约束条件	只有当之前已经选择了一个模块时才能被调用。
效果	启动所选模块的控制部分。控制部分应启动 TTCN-3 测试用例，如 TTCN-3 中所述。在执行控制部分的同时，TE 应为每个已启动和停止的测试用例调用提供的操作 `tciTestCaseStarted` 和 `tciTestCaseTerminated`。在终止控制部分后，TE 应调用提供的操作 `tciControlPartTerminated`。

7.3.1.1.10 tciStopControl

标记	`void tciStopControl()`
返回值	空
约束条件	只有当之前已经选择了一个模块时才能被调用。
效果	`tciStopControl` 停止执行控制部分。如果当前没有任何控制部分在执行，则操作应被忽略。如果已经直接启动一个测试用例，则应停止执行当前的测试用例，好像已经调用 `tciStopTestCase`。

7.3.1.2 提供的TCI-TM

本节规定了 TM 应提供给 TE 的操作。

7.3.1.2.1 tciTestCaseStarted

标记	`void tciTestCaseStarted(in TciTestCaseIdType testCaseId,`
	`in TciParameterListType parameterList,`
	`in TFloat timer)`	

输入参数	`testCaseId`	一个测试用例标识符，如 TTCN-3 模块中规定。
	`parameterList`	一个值清单，这些值是测试用例标记的一部分。当出现在 TTCN-3 测试用例声明中时，对参数清单中的各参数进行排序。
	`timer`	一个浮点数值，用于表示测试用例定时器的持续时间。
返回值	空	
约束条件	只有在已经使用要求的操作 `tciStartControl` 或 `tciStartTestCase` 启动了模块的控制部分或一个测试用例后才能被调用。	
效果	`tciTestCaseStarted` 向 `TM` 指明已经启动一个具有 `testCaseId` 的测试用例。应不区分测试用例是用要求的操作 `tciStartTestCase` 显性启动的还是在执行控制部分时隐性启动的。	

7.3.1.2.2 tciTestCaseTerminated

标记	`void tciTestCaseTerminated(in VerdictValue verdict,` `in TciParameterListType parameterList)`	
输入参数	`verdict`	测试用例的最终判定。
	`parameterList`	一个值清单，这些值是测试用例标记的一部分。当出现在 TTCN-3 测试用例声明中时，对参数清单中的各参数进行排序。
返回值	空	
约束条件	只有在已经使用要求的操作 `tciStartControl` 或 `tciStartTestCase` 启动了模块的控制部分或一个测试用例后才能被调用。	
效果	TE 应调用该操作来向测试管理器指明当前在 MTC 上执行的测试用例已终止，以及最终判定为 `verdict`。在调用一个 `tciTestCaseTerminated` 操作时，所有的 *out* 和 *inout* 测试用例参数都包含 `Value`。所有测试用例参数都包含特别值 `null`，原因是，它们只与测试用例启动相关，但不对调用做应答。	

7.3.1.2.3 tciControlTerminated

标记	`void tciControlTerminated()`
返回值	空
约束条件	只有当已经使用 `tciStartControl` 操作启动了模块执行后才能被调用。
效果	TE 应调用该操作来向测试管理器指明所选模块的控制部分刚刚终止执行。

7.3.1.2.4 tciGetModulePar

标记	`Value tciGetModulePar(in TciModuleParameterIdType parameterId)`	
输入参数	`parameterId`	模块参数的标识符，如 TTCN-3 中规定。
返回值	一个值	
约束条件	无论何时当 TE 需要访问模块参数的值时，应调用该操作。如果已经显性启动一个测试用例，则在一对 `tciStartTestCase` 和 `tciTestCaseTerminated` 之间，或者如果已经启动模块的控制部分，则在一对 `tciStartControl` 和 `tciControlTerminated` 之间，每个被访问的模块参数都应只被解析一次。	
效果	管理器向 TE 提供一个有关所指明 `parameterId` 的 `Value`。通过执行显性启动的测试用例，或者通过执行控制部分，每次调用 `tciGetModulePar()` 都应返回相同的值。如果管理器不能提供一个 TTCN-3 值，则应返回特别值 `null`。	

7.3.1.2.5 tciError

标记	void tciError(in TString message)	
输入参数	message	一个字符串值，即消息错误。
返回值	空	
约束条件	TE 可在任何时候调用该操作来指明一种不可修复的错误状况。这种错误状况可以通过 CH 或 CD 来指明，或者可能出现在 TE 中。	
效果	TE 指明出现了一种不可修复的错误状况。message 包含一个错误原因短语，可传送给测试系统用户。如果还在运行的话，测试管理器负责终止执行测试用例或控制部分的执行。测试管理器必须采取明确措施立即终止测试执行。	

7.3.2 TCI-CD接口

TCI 编码解码器接口（TCI-CD）描述要求可执行 TTCN-3 完成的各操作，以及某个编码方案的编码解码器实施方案应提供给 TE 的各操作（图 6）。

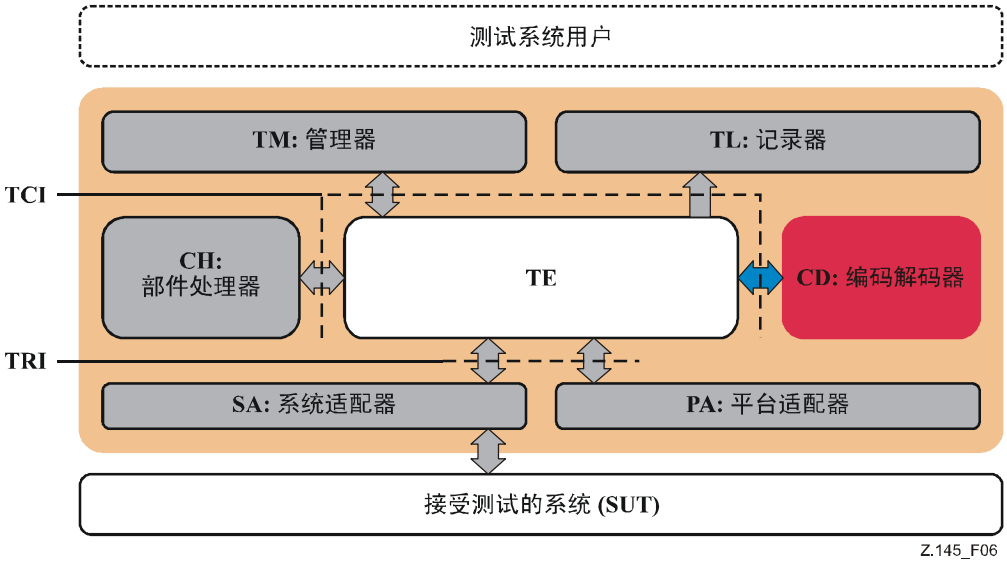

图 6/Z.145－TCI-CD接口

解码编码器实施方案依据编码属性把 TTCN-3 值编码为比特串，并依据解码假设对比特串进行解码。为了能够把比特串解码为 TTCN-3 值，CD 对 TE 提出了某些功能要求。反过来，CD 为可执行 TTCN-3 提供编码和解码功能。

第 10 节描述 TE 或 CD 操作调用的用法和顺序。

7.3.2.1 要求的TCI-CD

本节规定 CD 向 TE 要求的各操作。对 TCI-TM 和 TCI-CH 接口，也要求本节中规定的所有操作。

7.3.2.1.1 getTypeForName

标记	`Type getTypeForName(in TString typeName)`	
输入参数	`typeName`	指的是类型的 TTCN-3 名称，如 TTCN-3 模块中规定。以下为保留的类型名称，并应返回一个预定义的类型： “integer”（整数） “float”（浮点数） “bitstring”（比特串） “hexstring”（十六进制串） “octetstring”（八位字节串） “charstring”（字符串） “universal charstring”（通用字符串） “boolean”（布尔） “verdicttype”（判定类型） “objid”（对象标识符） typeName 须完全符合有关类型名称的要求，即 module.typeName。
返回值	一个类型，用于表示请求的 TTCN-3 类型。	
约束条件	---	
效果	返回一个表示 TTCN-3 类型的类型。可以使用有关预定义类型的 TTCN-3 关键字从 TE 获得预定义的 TTCN-3 类型。在这种情况下，`typeName` 表示基本的 TTCN-3 类型，如“charstring”、“bitstring”等。 如果不能返回所请求的类型，则返回特别值 `null`。注意：当把模块置为 `null` 时，不能获得 `anytype` 和地址。尽管它们是预定义的类型，但在各模块间它们可能是不同的。例如，地址可能是未经修改的预定义类型，或者是模块中的一个用户定义类型。不能对其它的预定义类型重新进行规定。	

7.3.2.1.2 getInteger

标记	`Type getInteger()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 整数类型。
效果	创建和返回一个基本的 TTCN-3 整数类型。

7.3.2.1.3 getFloat

标记	`Type getFloat()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 浮点数类型。
效果	创建和返回一个基本的 TTCN-3 浮点数类型。

7.3.2.1.4 getBoolean

标记	`Type getBoolean()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 布尔类型。
效果	创建和返回一个基本的 TTCN-3 布尔类型。

7.3.2.1.5 getObjid

标记	`Type getObjid()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 对象标识符类型。
效果	创建和返回一个基本的 TTCN-3 对象标识符类型。

7.3.2.1.6 getCharstring

标记	`Type getCharstring()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 字符串类型。
效果	创建和返回一个基本的 TTCN-3 字符串类型。

7.3.2.1.7 getUniversalCharstring

标记	`Type getUniversalCharstring()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 通用字符串类型。
效果	创建和返回一个基本的 TTCN-3 通用字符串类型。

7.3.2.1.8 getHexstring

标记	`Type getHexstring()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 十六进制串类型。
效果	创建和返回一个基本的 TTCN-3 十六进制串类型。

7.3.2.1.9 getBitstring

标记	`Type getBitstring()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 比特串类型。
效果	创建和返回一个基本的 TTCN-3 比特串类型。

7.3.2.1.10 getOctetstring

标记	`Type getOctetstring()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 八位字节串类型。
效果	创建和返回一个基本的 TTCN-3 八位字节串类型。

7.3.2.1.11 getVerdict

标记	`Type getVerdict()`
返回值	`Type` 的一个实例，用于表示一个 TTCN-3 判定类型。
效果	创建和返回一个基本的 TTCN-3 判定类型。

7.3.2.1.12 tciErrorReq

标记	`void tciErrorReq(in TString message)`	
输入参数	`message`	一个字符串值，即用于描述问题的错误短语。
返回值	空	
约束条件	无论何时当出现某种错误状况时都调用该操作。	
效果	TE 应被告知有关 CD 内出现的不可修复错误状况，并把错误指示传送给测试管理器。	

7.3.2.2 提供的TCI-CD

本节规定了 TM 应提供给 TE 的操作。

7.3.2.2.1 解码

标记	`Value decode(in TriMessageType message, in Type decodingHypothesis)`	
输入参数	`message`	待解码的编码消息。
	`decodingHypothesis`	解码可依据的假设。
返回值	如果值是一种兼容的类型，如 `decodingHypothesis`，则返回经过解码的值，否则返回特别值 `null`。	
约束条件	无论何时当 TE 需要对一个经过编码的值进行解码时，应调用该操作。在收到一个经过编码的值后，TE 可以立即进行解码，或者出于性能方面的考虑，可推迟解码，直至实际访问经过编码的值。	
效果	该操作依据编码规则对消息进行解码，并返回一个 TTCN-3 值。应使用 `decodingHypothesis` 来确定经过编码的值能否解码。如果一个编码规则不是自充分的，即如果经过编码的消息未内在地包含其类型，则应使用 `decodingHypothesis`。如果无需解码假设就能对经过编码的值进行解码，则若取决于经过编码消息的类型不兼容于解码假设，则返回特别值 `null`。	

7.3.2.2.2 编码

标记	`TriMessageType encode(in Value value)`	
输入参数	`value`	待编码的值。
返回值	为规定的编码规则返回一个经过编码的 `TriMessage`。	
约束条件	无论何时当 TE 需要对一个 Value 进行编码时，应调用该操作。	
效果	依据编码规则，返回一个经过编码的 `TriMessage`。	

7.3.3 TCI-CH接口

TCI 部件处理接口（TCI-CH）描述要求可执行 TTCN-3 完成的各操作，以及部件处理实施方案应提供给 TE 的各操作（图 7）。

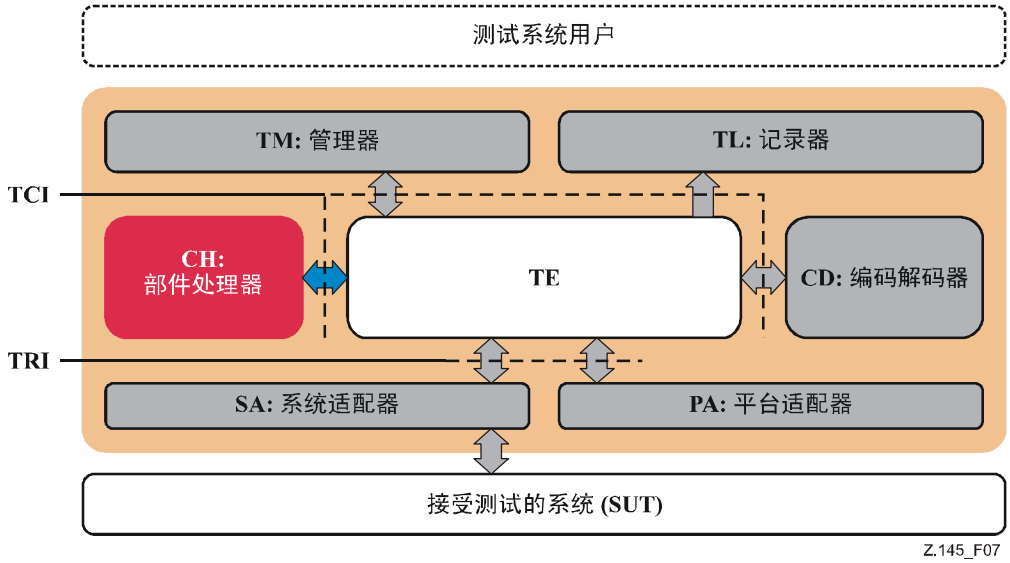

图 7/Z.145－TCI-CH接口

部件处理实施方案在一个或多个参与测试会议的可执行 TTCN-3 间分发 TTCN-3 配置操作，如 `create`、`connect` 和 `start`，以及内部部件通信，如在一个连接端口上的 `send`。注意：尽管 TE 的多个实例可以参与一个测试会议，但这不是强制性的。

基本原则是 TCI-CH 不执行任何种类的 TTCN-3 功能。相反地，TE 应告知它以下信息，例如应创建一个测试部件。基于部件处理器（CH）的内部知识，创建一个测试部件的请求应被传送给另一个（远程的）参与会议的 TE。这个第二个（远程的）参与会议的 TE 应创建 TTCN-3 部件，并回传一个句柄给提出请求的（本地的）TE。提出请求的（本地的）TE 现通过该部件句柄在所建的测试部件上进行操作。

在操作定义内，术语“本地的 TE”和“远程的 TE”用于强调以下事实，即一个测试系统实施方案可以分布于若干个测试设备上，当中的每一个测试设备作为一个完整 TE 的宿主。术语“本地的”和“远程的”总是指当前所述的接口。为方便起见，术语“本地的”总是指作为操作被调用者的 TE（对要求的操作）或者作为操作调用者的 TE（对提供的操作）。概念上，认为 TE 是分布式的，CH 是非分布式的。这可以通过使用一个集中式体系结构或者使用一个抽象自分布概念的中间件平台来实现。尽管 TE 可以分布于不同的物理设备上，但可能存在这样的配置，即当中只有一个非分布式 TE 可以参与测试会议。在这种情况下，术语“本地的”和“远程的”指的是同一 TE 实例。

第 10 条描述 TE 或 CH 操作调用的用法和顺序。

尽管参与测试会议的所有可执行 TTCN-3 都是相等的，但存在一个特别的 TE*。该 TE*是这样一个 TE，即当中显性的 `tciStartTestCase()` 或 `tciStartControl()`已经过处理。存在这种差别的理由是该 TE* 应计算全局判定。一旦测试执行终止，TE*应告知测试管理器，而后提供测试用例的全局判定。

7.3.3.1 要求的TCI-CH

本节规范 CH 对 TE 要求的操作。除了在本节中规范的操作，还需要 TCI-CD 接口的所有要求的操作。

7.3.3.1.1 tciEnqueueMsgConnected

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`receiver`	接收部件的标识符。
	`rcvdMessage`	要排队的值。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciSendConnected` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 排队等待接收到的值进入所指示的接收机部件的本地端口队列。	

7.3.3.1.2 tciEnqueueCallConnected

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`receiver`	接收部件的标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciCallConnected` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。所有 *in* 和 *inout* 程序参数都包括值。所有 *out* 程序参数应包括特别值 `null`，因为他们仅与应答程序调用而不是程序调用本身相关。 程序参数是 TTCN-3 标记模板中规定的参数。	
效果	TE 排队等待所指示的接收机部件的本地端口队列上的调用。	

7.3.3.1.3 tciEnqueueReplyConnected

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

输入参数	`sender`	发送应答的端口标识符。
	`receiver`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
	`returnValue`	程序调用的（选用）返回值。
返回值	`void`	
约束条件	当远端 TE 上的一个*提供的* `tciReplyConnected` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。所有 *out* 和 *inout* 程序参数以及返回值都包括值。所有 *in* 程序参数应包括特别值 `null`，因为他们仅与程序调用而不是应答调用相关。`parameterList` 包括程序调用参数。这些参数是 TTCN-3 标记模板中规定的参数。如果在 TTCN-3 ATS 中没有为程序标记规定返回类型，则为 `returnValue` 应通过特别值 `null`。	
效果	TE 排队等待所指示的接收机部件的本地端口队列上的应答。	

7.3.3.1.4 tciEnqueueRaiseConnected

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

输入参数	`sender`	发送应答的端口标识符。
	`receiver`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`exception`	例外情况。
返回值	`void`	
约束条件	当远端 TE 上的一个*提供的* `tciRaiseConnected` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 排队等待所指示的接收机部件的本地端口队列上的例外情况。	

7.3.3.1.5 tciCreateTestComponent

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

输入参数	`kind`	应创建的部件种类，`MTC`、`PTC` 或 `CONTROL`。
	`componentType`	应创建 TTCN-3 部件类型的标识符。
	`name`	应创建部件的名称。
返回值	用于创建部件的 `TriComponentIdType` 值。	
约束条件	当远端 TE 上的一个*提供的* *tciCreateTestComponentReq* 已经被调用，则应在本地 TE 上由 CH 调用该项操作。`componentType` 应被设置为特别值 `null`，如果创建了种类控制的测试部件的话。`name` 应被设置为特别值 `null`，如果在 TTCN-3 创建声明中没有给出名称的话。	
效果	TE 创建了 `componentType` 的一个 TTCN-3 测试部件并把一个 `TriComponentIdType` 参考传回给 CH。CH 把参考发回远端 TE 联系。	

7.3.3.1.6 tciStartTestComponent

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

输入参数	`component`	要启动的部件标识符。参考由 `tciCreateTestComponent` 调用原来创建的一个标识符。
	`behaviour`	在部件上要启动的行为标识符。
	`parameterList`	值的列表，其中每个值规定了一个参数，该参数来自于正在启动功能的 TTCN-3 功能声明中规定的参数列表。在 `parameterList` 中的参数按他们出现在测试用例的 TTCN-3 标记中的顺序排序。如果没有参数必须被通过，则应通过 `null` 值或一个空的 `parameterList`，即长度为 0 的列表。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciStartTestComponentReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。由于正在启动的功能只允许有 *in* 参数（ITU-T Z.140 建议书 [2]），`parameterList` 只包括 *in* 参数。	
效果	TE 应在指示部件上启动指示的行为。	

7.3.3.1.7 tciStopTestComponent

标记	`void tciStopTestComponent(in TriComponentIdType component)`	
输入参数	`component`	要停止的部件标识符。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciStopTestComponentReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 应在指示部件上停止指示的行为。	

7.3.3.1.8 tciConnect

标记	`void tciConnect(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
输入参数	`fromPort`	从连接到的测试部件端口来的标识符。
	`toPort`	被连接到的测试部件端口的标识符。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciConnect` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 应把指示端口连接到其他端口。	

7.3.3.1.9 tciDisconnect

标记	`void tciDisconnect(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
输入参数	`fromPort`	要断开的测试部件端口的标识符。
	`toPort`	要断开的测试部件端口的标识符。
返回值	`void`	
约束条件	当远端 TE 上的一个提供的 `tciDisconnect` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 应断开指示端口。	

7.3.3.1.10 tciMap

标记	`void tciMap(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	来自映射的测试部件端口的标识符。
	`toPort`	要映射到的测试部件端口的标识符。
返回值	`void`	
约束条件	当远端 TE 上的一个*提供的* `tciMapReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 应把指示端口映射到其他端口。	

7.3.3.1.11 tciUnmap

标记	`void tciUnmap(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	不被映射的测试部件端口的标识符。
	`toPort`	不被映射的测试部件端口的标识符。
返回值	`void`	
约束条件	当远端 TE 上的一个*提供的* `tciUnmapReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	TE 应不映射指示端口。	

7.3.3.1.12 tciTestComponentTerminated

标记	`void tciTestComponentTerminated(in TriComponentIdType component, in VerdictValue verdict)`	
输入参数	`component`	已被终止部件的标识符。
	`verdict`	终止部件后判断。
返回值	`void`	
约束条件	当远端 TE 上的一个*提供的* `tciTestComponentTerminatedReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	通知本地 TE 在远端 TE 上终止所指示的测试部件。由于在测试部件上正在执行的功能只能有 *in* 参数（ITU-T Z.140 建议书 [2]），`tciTestComponentTerminated` 操作没有一个 `parameterList` 参数。	

7.3.3.1.13 tciTestComponentRunning

标记	`TBoolean tciTestComponentRunning(in TriComponentIdType component)`	
输入参数	`component`	要进行运行检查的部件的标识符。
返回值	如果所指示的部件仍在执行一个行为，则为 `true`，否则为 `false`。	
约束条件	当远端 TE 上的一个*提供的* `tciTestComponentRunningReq` 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	本地 TE 确定所指示的部件是否在执行一个测试行为。如果部件在执行行为，则返回 `true`。在任何其他情况下，例如，测试部件已经完成执行，或还没有启动测试部件等，则返回 `false`。在操作返回后，CH 应把该值发回远端 TE 联系。	

7.3.3.1.14 tciTestComponentDone

标记	TBoolean tciTestComponentDone(in TriComponentIdType comp)	
输入参数	comp	要进行完成检查的部件的标识符。
返回值	如果所指示的部件已经完成执行行为，则为 true，否则为 false。	
约束条件	当远端 TE 上的一个*提供的* tciTestComponentDoneReq 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	本地 TE 确定所指示的部件是否已经完成测试行为。如果部件已经完成，则返回 true。在任何其他情况下，例如，还没有启动测试部件，或测试部件仍在执行，则返回 false。在操作返回后，CH 应把该值发回远端 TE 联系。	

7.3.3.1.15 tciGetMTC

标记	TriComponentIdType tciGetMTC()
返回值	如果 MTC 在本地 TE 上执行，则为 MTC 的 TriComponentIdType 值，否则为特别值 null。
约束条件	当远端 TE 上的一个*提供的* tciGetMTCReq 已经被调用，则应在本地 TE 上由 CH 调用该项操作。
效果	本地 TE 确定 MTC 是否在本地 TE 上执行。如果 MTC 在本地 TE 上执行，则返回 MTC 的部件标识符。如果 MTC 不在本地 TE 上执行，则返回特别值 null。该操作对 MTC 的执行没有影响。在操作返回后，CH 应把该值发回远端 TE 联系。

7.3.3.1.16 tciExecuteTestCase

标记	void tciExecuteTestCase(in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)	
输入参数	testCaseId	正如 TTCN-3 模块中规定的测试用例标识符。
	tsiPortList	包括在测试用例系统部件的定义中公布的所有端口，即 TSI 端口。如果没有为测试用例明确地规定系统部件，则 tsiPortList 包括所有 MTC 的通信端口。tsiPortList 内的端口按他们出现在相应的 TTCN-3 部件类型声明上的顺序排序。如果没有端口必须被通过，则应通过 null 值或一个空的 tsiPortList，即长度为 0 的列表。
返回值	void	
约束条件	当远端 TE 上的一个*提供的* tciExecuteTestCaseReq 已经被调用，则应在相应的本地 TE 上由 CH 调用该项操作。	
效果	本地 TE 确定是否应该完成到 SUT 的静态连接以及 TSI 端口通信手段的初始化。	

7.3.3.1.17 tciReset

标记	void tciReset()
返回值	void
约束条件	当远端 TE 上的一个*提供的* tciResetReq 已经被调用，则应在相应的本地 TE 上由 CH 调用该项操作。
效果	TE 能够决定采取任何方式在本地重置测试系统。

7.3.3.1.18 tciKillTestComponent

标记	void tciKillTestComponent(in TriComponentIdType comp)	
输入参数	comp	要被取消的部件的标识符。
返回值	void	
约束条件	当远端 TE 上的一个*提供的* tciKillTestComponentReq 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	如果有必要，TE 会停止在所指示的部件上的行为并转到被取消的状态。	

7.3.3.1.19 tciTestComponentAlive

标记	TBoolean tciTestComponentAlive(in TriComponentIdType comp)	
输入参数	comp	处于激活状态的待检测部件的标识符。
返回值	如果所指示的部件激活，则为 true，否则为 false。	
约束条件	当远端 TE 上的一个*提供的* tciTestComponentAliveReq 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	本地 TE 确定所指示的部件是否被激活。在操作返回后，CH 应把该值发回远端 TE 联系。	

7.3.3.1.20 tciTestComponentKilled

标记	TBoolean tciTestComponentKilled(in TriComponentIdType comp)	
输入参数	comp	处于被取消状态的待检测部件的标识符。
返回值	如果所指示的部件已经被取消，则为 true，否则为 false。	
约束条件	当远端 TE 上的一个*提供的* tciTestComponentKilledReq 已经被调用，则应在本地 TE 上由 CH 调用该项操作。	
效果	本地 TE 确定所指示的部件是否处于被取消的状态。如果是，则返回 true。在任何其他情况下，返回 false。在操作返回后，CH 应把该值发回远端 TE 联系。	

7.3.3.2 提供的TCI-CH

本节规定了 CH 应提供给 TE 的操作。

7.3.3.2.1 tciSendConnected

标记	void tciSendConnected(in TriPortIdType sender, in TriComponentIdType receiver, in Value sendMessage)	
输入参数	sender	经由消息发送的发送部件的端口标识符。
	receiver	接收部件的标识符。
	sendMessage	要发送的消息。
返回值	void	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 单播发送操作时，应由 TE 调用该操作。	
效果	仅把异步传送发送到给定接收机部件上。CH 把消息发送到远端 TE 正在运行的接收机上，并让数据在远端 TE 内排队。	

7.3.3.2.2 tciSendConnectedBC

标记	void tciSendConnectedBC(in TriPortIdType sender, in Value sendMessage)	
输入参数	sender	经由消息发送的发送部件的端口标识符。
	sendMessage	要发送的消息。
返回值	void	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 广播发送操作时，应由 TE 调用该操作。	
效果	仅把异步传送发送到连接到该端口的所有部件上。CH 把消息发送到所有远端 TE 正在运行的接收机上，并让数据在远端 TE 内排队。	

7.3.3.2.3 tciSendConnectedMC

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`receivers`	接收部件的标识符。
	`sendMessage`	要发送的消息。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 多播发送操作时，应由 TE 调用该操作。	
效果	仅把异步传送发送到所有给定接收机部件上。CH 把消息发送到所有远端 TE 正在运行的接收机上，并让数据在远端 TE 内排队。	

7.3.3.2.4 tciCallConnected

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`receiver`	接收部件的标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 单播调用操作时，应由 TE 调用该操作。所有 *in* 和 *inout* 程序参数都包括值。 所有 *out* 程序参数应包括特别值 `null`，因为他们仅与应答程序调用而不是程序调用本身相关。这些程序参数是 TTCN-3 标记模板中规定的参数。	
效果	调用该操作时，TE 能够启动对应于被调用部件接收机处的标记标识符 `signature` 的程序调用。`tciCallConnected` 操作应返回，不用等待发出程序调用的返回。注意到在 `tciCallConnected` 操作标记中没有包括选用的超时值，可以在 TTCN-3 ATS 中为调用操作规定超时值。TE 负责通过在带有单独 TRI 操作调用的 PA 内为 TTCN-3 调用操作启动一个定时器，即 `triStartTimer`，来说明该问题。CH 把调用发送到所有远端 TE 正在运行的接收机上，并让调用在远端 TE 内排队。	

7.3.3.2.5 tciCallConnectedBC

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 广播调用操作时，应由 TE 调用该操作。所有 *in* 和 *inout* 程序参数都包括值。所有 *out* 程序参数应包括特别值 `null`，因为他们仅与应答程序调用而不是程序调用本身相关。这些程序参数是 TTCN-3 标记模板中规定的参数。	
效果	调用该操作时，TE 能够启动对应于被调用部件接收机处的标记标识符 `signature` 的程序调用。`tciCallConnected` 操作应返回，不用等待发出程序调用的返回。注意到在 `tciCallConnected` 操作标记中没有包括选用的超时值，可以在 TTCN-3 ATS 中为调用操作规定超时值。TE 负责通过在带有单独 TRI 操作调用的 PA 内为 TTCN-3 调用操作启动一个定时器，即 `triStartTimer`，来说明该问题。CH 把调用发送到所有远端 TE 正在运行的接收机上，并让调用在远端 TE 内排队。	

7.3.3.2.6 tciCallConnectedMC

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

输入参数	`sender`	经由消息发送的发送部件的端口标识符。
	`receivers`	接收部件的标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 多播调用操作时，应由 TE 调用该操作。所有 *in* 和 *inout* 程序参数都包括值。所有 *out* 程序参数应包括特别值 `null`，因为他们仅与应答程序调用而不是程序调用本身相关。这些程序参数是 TTCN-3 标记模板中规定的参数。	
效果	调用该操作时，TE 能够启动对应于被调用部件接收机处的标记标识符 `signature` 的程序调用。`tciCallConnected` 操作应返回，不用等待发出程序调用的返回。注意到在 `tciCallConnected` 操作标记中没有包括选用的超时值，可以在 TTCN-3 ATS 中为调用操作规定超时值。TE 负责通过在带有单独 TRI 操作调用的 PA 内为 TTCN-3 调用操作启动一个定时器，即 `triStartTimer`，来说明该问题。CH 把调用发送到所有远端 TE 正在运行的接收机上，并让调用在远端 TE 内排队。	

7.3.3.2.7 tciReplyConnected

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

输入参数	`sender`	发送应答的端口标识符。
	`receiver`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
	`returnValue`	程序调用的（选用）返回值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 单播应答操作时，应由 TE 调用该操作。 所有 *out* 和 *inout* 程序参数以及返回值都包括值。 所有 *in* 程序参数应包括特别值 `null`，因为他们仅与程序调用而不是程序调用的应答相关。`parameterList` 包括程序调用参数。这些参数是 TTCN-3 标记模板中规定的参数。如果在 TTCN-3 ATS 中没有为程序标记规定返回类型，则应为返回值通过特别值 `null`。	
效果	调用该操作时，CH 能够发送应答至对应于标记标识符 `signature` 和部件标识符 `receiver` 的程序调用。CH 把应答发送到所有远端 TE 正在运行的接收机上，并让应答在远端 TE 内排队。	

7.3.3.2.8 tciReplyConnectedBC

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

输入参数	`sender`	发送应答的端口标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	参数值的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
	`returnValue`	程序调用的（选用）返回值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 广播应答操作时，应由 TE 调用该操作。 所有 *out* 和 *inout* 程序参数以及返回值都包括值。 所有 *in* 程序参数应包括特别值 `null`，因为他们仅与程序调用而不是程序调用的应答相关。`parameterList` 包括程序调用参数。这些参数是 TTCN-3 标记模板中规定的参数。如果在 TTCN-3 ATS 中没有为程序标记规定返回类型，则应为返回值通过特别值 `null`。	
效果	调用该操作时，CH 能够发送应答至对应于标记标识符 `signature` 和所有连接到 `sender` 的部件的程序调用。CH 把例外情况发送到所有远端 TE 正在运行的接收机上，并让例外情况在远端 TE 内排队。	

7.3.3.2.9 tciReplyConnectedMC

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

输入参数	`sender`	发送应答的端口标识符。
	`receivers`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`parameterList`	编码参数的列表是所指示的标记的一部分。在 `parameterList` 中的参数按他们出现在 TTCN-3 标记声明中的顺序排序。
	`returnValue`	程序调用的（选用）返回值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 多播应答操作时，应由 TE 调用该操作。 所有 *out* 和 *inout* 程序参数以及返回值都包括值。所有 *in* 程序参数应包括特别值 `null`，因为他们仅与程序调用而不是程序调用的应答相关。`parameterList` 包括程序调用参数。这些参数是 TTCN-3 标记模板中规定的参数。如果在 TTCN-3 ATS 中没有为程序标记规定返回类型，则应为返回值通过特别值 `null`。	
效果	调用该操作时，CH 能够发送应答至对应于标记标识符 `signature` 和 `receiver` 中的其中一个部件标识符的程序调用。CH 把应答发送到远端 TE 正在运行的接收机上，并让应答在远端 TE 内排队。	

7.3.3.2.10 tciRaiseConnected

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

输入参数	`sender`	发送应答的端口标识符。
	`receiver`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`exception`	例外值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 单播引起操作时，应由 TE 调用该操作。	
效果	调用该操作时，CH 能够引起一个例外情况至对应于标记标识符 `signature` 和部件标识符 `receiver` 的程序调用。 CH 把例外情况发送到远端 TE 正在运行的接收机上，并让例外情况在远端 TE 内排队。	

7.3.3.2.11 tciRaiseConnectedBC

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

输入参数	`sender`	发送应答的端口标识符。
	`signature`	程序调用标记的标识符。
	`exception`	例外值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 广播引起操作时，应由 TE 调用该操作。	
效果	调用该操作时，CH 能够引起一个例外情况至对应于标记标识符 `signature` 和所有连接到 `sender` 的部件的程序调用。CH 把例外情况发送到所有远端 TE 正在运行的接收机上，并让例外情况在远端 TE 内排队。	

7.3.3.2.12 tciRaiseConnectedMC

标记	`void tciRaiseConnectedMC(in TriPortIdType sender,`
	`in TriComponentIdListType receiver,`
	`in TriSignatureIdType signature,`
	`in Value exception)`	

输入参数	`sender`	发送应答的端口标识符。
	`receivers`	接收应答的部件标识符。
	`signature`	程序调用标记的标识符。
	`exception`	例外值。
返回值	`void`	
约束条件	当在一个已被连接到其他部件端口的端口上执行 TTCN-3 多播引起操作时，应由 TE 调用该操作。	
效果	调用该操作时，CH 能够引起一个例外情况至对应于标记标识符 `signature` 和部件标识符 `receiver` 中的其中一个的程序调用。CH 把例外情况发送到所有远端 TE 正在运行的接收机上，并让例外情况在远端 TE 内排队。	

7.3.3.2.13 tciCreateTestComponentReq

标记	`TriComponentIdType tciCreateTestComponentReq(in TciTestComponentKindType kind, in Type componentType, in TString name)`	
输入参数	`kind`	应创建的部件种类，`MTC`、`PTC` 或 `CONTROL`。
	`componentType`	应创建 TTCN-3 部件类型的标识符。
返回值	用于所创建部件的 `TriComponentIdType` 值。	
约束条件	当必须要创建部件时，应从 TE 调用该操作，如果 TTCN-3 创建操作被调用则为显性调用，如果必须要创建主测试部件（MTC）或一个控制部件则为隐性调用。如果在 TTCN-3 创建声明中没有给出名称，则 name 应被置为特别值 null。	
效果	CH 把部件创建请求发送给远端 TE 并调用那里的 `tciCreateTestComponent` 操作以获得该部件的部件标识符。	

7.3.3.2.14 tciStartTestComponentReq

标记	`void tciStartTestComponentReq(in TriComponentIdType component, in TciBehaviourIdType behaviour, in TciParamaterListType parameterList)`	
输入参数	`component`	要启动的部件标识符。
	`behaviour`	在部件上要启动的行为标识符。
	`parameterList`	值的列表，其中每个值规定了一个参数，该参数来自于正在启动功能的 TTCN-3 功能声明中规定的参数列表。在 `parameterList` 中的参数按他们出现在 TTCN-3 测试用例标记中的顺序排序。如果没有参数必须被通过，则应通过 `null` 值或一个空的 `parameterList`，即长度为 0 的列表。
返回值	`void`	
约束条件	当执行 TTCN-3 启动操作时，应由 TE 调用该操作。对于正在启动的功能只允许有 *in* 参数（ITU-T Z.140 建议书 [2]），因此 `parameterList` 只包括 *in* 参数。	
效果	CH 把启动部件请求发送给远端 TE 并调用那里的 `tciStartTestComponent` 操作。	

7.3.3.2.15 tciStopTestComponentReq

标记	`void tciStopTestComponentReq(in TriComponentIdType component)`	
输入参数	`component`	要停止的部件标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 停止操作时，应由 TE 调用该操作。	
效果	CH 把停止部件请求发送给远端 TE 并调用那里的 `tciStopTestComponent` 操作。	

7.3.3.2.16 tciConnectReq

标记	`void tciConnectReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	从连接到的测试部件端口来的标识符。
	`toPort`	被连接到的测试部件端口的标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 连接操作时，应由 TE 调用该操作。	
效果	CH 把连接请求发送给远端 TE，调用那里的 `tciConnect` 操作以在两个所指示的端口之间建立一个逻辑连接。注意到两个端口可能都在远端 TE 上。在这种情况下，仅在调用两个远端 TE 上的 `tciConnect` 操作之后，该操作返回。	

7.3.3.2.17 tciDisconnectReq

标记	`void tciDisconnectReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	要断开的测试部件端口的标识符。
	`toPort`	要断开的测试部件端口的标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 断开连接操作时，应由 TE 调用该操作。	
效果	CH 把断开连接请求发送给远端 TE，调用那里的 `tciDisconnect` 操作以在两个所指示的端口之间断开逻辑连接。注意到两个端口可能都在远端 TE 上。在这种情况下，仅在调用两个远端 TE 上的 `tciDisconnect` 操作之后，该操作返回。	

7.3.3.2.18 tciMapReq

标记	`void tciMapReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	来自映射的测试部件端口的标识符。
	`toPort`	要映射到的测试部件端口的标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 映射操作时，应由 TE 调用该操作。	
效果	CH 把映射请求发送给远端 TE，调用那里的 `tciMap` 操作以在两个所指示的端口之间建立一个逻辑连接。	

7.3.3.2.19 tciUnmapReq

标记	`void tciUnmapReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
输入参数	`fromPort`	不被映射的测试部件端口的标识符。
	`toPort`	不被映射的测试部件端口的标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 取消映射操作时，应由 TE 调用该操作。	
效果	CH 把取消映射请求发送给远端 TE，调用那里的 `tciUnmap` 操作以在两个所指示的端口之间断开逻辑连接。	

7.3.3.2.20 tciTestComponentTerminatedReq

标记	`void tciTestComponentTerminatedReq(in TriComponentIdType component, in VerdictValue verdict)`	
输入参数	`component`	已被终止部件的标识符。
	`verdict`	终止部件后判断。
返回值	`void`	
约束条件	当测试部件终止执行时，该操作应由 TE 调用，如果 TTCN-3 停止操作，则为显性调用，如果已经到达最后声明，则为隐性调用。	
效果	通知 CH 终止所指示的测试部件。由于在测试部件上正执行的功能只有 *in* 参数（ITU-T Z.140 建议书 [2]），所以 `tciTestComponentTerminateReq` 操作没有 `parameterList` 参数。CH 就所指示部件的终止，与所有参与的 TE 和追踪整个判定过程的特殊 TE*沟通。	

7.3.3.2.21 tciTestComponentRunningReq

标记	`TBoolean tciTestComponentRunningReq(in TriComponentIdType component)`	
输入参数	`component`	要进行运行检查的部件的标识符。
返回值	如果所指示的部件仍在执行一个行为，则为 `true`，否则为 `false`。	
约束条件	当执行 TTCN-3 运行操作时，应由 TE 调用该操作。	
效果	CH 把运行请求发送到远端 TE，让测试部件被检测，此处调用 `tciTestComponentRunning` 操作来检测所指示测试部件的执行状态。	

7.3.3.2.22 tciTestComponentDoneReq

标记	`TBoolean tciTestComponentDoneReq(in TriComponentIdType comp)`	
输入参数	`comp`	要进行完成检查的部件的标识符。
返回值	如果所指示部件已经完成执行行为，则为 `true`；否则为 `false`。	
约束条件	当执行 TTCN-3 取消完成操作时，应由 TE 调用该操作。	
效果	CH 把完成请求发送到远端 TE，让测试部件被检测，此处调用 `tciTestComponentDone` 操作来检测所指示测试部件的执行状态。	

7.3.3.2.23 tciGetMTCReq

标记	`TriComponentIdType tciGetMTCReq()`
返回值	MTC 的 `TriComponentIdType` 值。
约束条件	当执行 TTCN-3 mtc 操作时，应由 TE 调用该操作。
效果	CH 确定 MTC 的部件标识符。

7.3.3.2.24 tciExecuteTestCaseReq

标记	`void tciExecuteTestCaseReq(in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`	
输入参数	`testCaseId`	正如 TTCN-3 模块中规定的测试用例标识符。
	`tsiPortList`	`tsiPortList` 包括在测试用例系统部件的定义中公布的所有端口，即 TSI 端口。如果为测试用例明确规定系统部件，则 `tsiPortList` 包括所有 MTC 的通信端口。 在 `tsiPortList` 中的端口按他们出现在相应TTCN-3部件类型声明中的顺序排序。如果没有端口必须被通过，则应通过 `null` 值或一个空的 `tsiPortList`，即长度为 0 的列表。
返回值	`void`	
约束条件	在 MTC 上启动测试用例行为之前（在 TTCN-3 执行操作期间），该操作能立即被 TE 调用。	
效果	CH 把执行测试用例的请求发送到远端 TE，该远端 TE 有所指示测试用例的系统端口。能够建立到 SUT 的静态连接以及用于 TSI 端口的通信初始化的方式。	

7.3.3.2.25 tciResetReq

标记	`void tciResetReq()`
返回值	`void`
约束条件	该操作能在任何时候被 TE 调用以重置测试系统。
效果	CH 把重置请求发送给所有涉及的 TE。

7.3.3.2.26 tciKillTestComponentReq

标记	`void tciKillTestComponentReq(in TriComponentIdType comp)`	
输入参数	`comp`	要被取消的部件的标识符。
返回值	`void`	
约束条件	当执行 TTCN-3 取消操作时，应由 TE 调用该操作。	
效果	CH 把取消部件请求发送给远端 TE 并在那里调用 `tciKillTestComponent` 操作。	

7.3.3.2.27 tciTestComponentAliveReq

标记	`TBoolean tciTestComponentAliveReq(in TriComponentIdType comp)`	
输入参数	`comp`	处于激活状态的待检测部件的标识符。
返回值	如果所指示部件激活，则为 `true`；否则为 `false`。	
约束条件	当执行 TTCN-3 激活操作时，应由 TE 调用该操作。	
效果	CH 把请求发送给远端 TE，创建需讨论的测试部件，并在那里调用 `tciTestComponentAlive` 操作以检测所指示测试部件的状态。	

7.3.3.2.28 tciTestComponentKilledReq

标记	`TBoolean tciTestComponentKilledReq(in TriComponentIdType comp)`	
输入参数	`comp`	处于取消状态的待检测部件的标识符。
返回值	如果所指示部件取消，则为 `true`；否则为 `false`。	
约束条件	当执行 TTCN-3 取消操作时，应由 TE 调用该操作。	
效果	CH 把请求发送给远端 TE，创建需讨论的测试部件，并在那里调用 `tciTestComponentKilled` 操作以检测所指示测试部件的状态。	

7.3.4 TCI-TL接口

TCI 测试记录接口（TCI-TL）描述了要求实施可执行的 TTCN-3 操作以及测试记录实施应提供给 TE 的操作（图 8）。

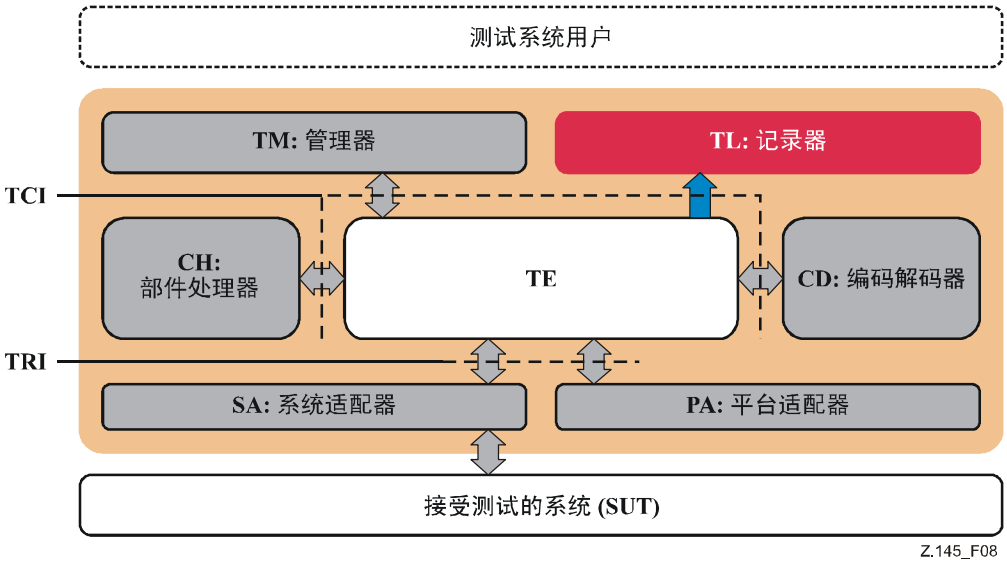

图 8/Z.145 —TCI-TL接口

记录为所有 TTCN-3 级别的操作提供一个操作，把正由 TE、SA、PA、CH 或 CD 处理的相应事件记录到用户上。

7.3.4.1 提供的TCI-TL

本节规定了 TL 应提供给 TE 的操作。

7.3.4.1.1 tliTcExecute

标记	void tliTcExecute(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	tcId	待执行的测试用例。
	pars	测试用例要求的参数列表。
	dur	执行持续的时间。
返回值	void	
约束条件	应由 TE 调用以记录执行测试用例的请求。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.2 tliTcStart

标记	void tliTcStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	tcId	待执行的测试用例。
	pars	测试用例要求的参数列表。
	dur	执行持续的时间。
返回值	void	
约束条件	应由 TE 调用以记录测试用例的启动。该事件在启动测试用例之前发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.3 tliTcStop

标记	void tliTcStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录测试用例的停止。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.4 tliTcStarted

标记	void tliTcStarted(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	tcid	待执行的测试用例。
	pars	测试用例要求的参数列表。
	dur	执行持续的时间。
返回值	void	
约束条件	应由 TM 调用以记录测试用例的启动。该事件在启动测试用例之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.5 tliTcTerminated

标记	void tliTcTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in VerdictValue outcome)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	tcId	待执行的测试用例。
	pars	测试用例要求的参数列表。
	outcome	测试用例的判定。
返回值	void	
约束条件	应由 TM 调用以记录测试用例的终止。该事件在测试用例终止之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.6 tliCtrlStart

标记	void tliCtrlStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录控制部分的启动。该事件在启动控制之前发生。如果该控制不由 TRI 部件表示，则 c 为 null。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.7 tliCtrlStop

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

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
返回值	`void`	
约束条件	应由 TE 调用以记录控制部分的停止。该事件在停止控制之前发生。如果该控制不由 TRI 部件表示，则 c 为 null。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.8 tliCtrlTerminated

标记	`void tliCtrlTerminated(in TString am, in TInteger ts,`
	`in TString src, in TInteger line,`
	`in TriComponentIdType c)`	

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
返回值	`void`	
约束条件	应由 TM 调用以记录控制部分的终止。该事件在控制终止之后发生。如果控制不由 TRI 部件表示，则 c 为 null。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.9 tliMSend_m

标记	`void tliMSend_m(in TString am, in TInteger ts,`
	`in TString src, in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port, in Value msgValue,`
	`in TriAddressType address,`
	`in TriStatusType encoderFailure,`
	`in TriMessageType msg,`
	`in TriStatusType transmissionFailure)`	

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送消息经由的端口。
	`msgValue`	待编码和发送的值。
	`address`	在 SUT 内目的地地址。
	`encoderFailure`	在编码时可能会失败的消息。
	`msg`	已编码的消息。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 SA 调用以记录单播发送操作。该事件在发送之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.10 tliMSend_m_BC

标记	void tliMSend_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送消息经由的端口。
	msgValue	待编码和发送的值。
	encoderFailure	在编码时可能会失败的消息。
	msg	已编码的消息。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录广播发送操作。该事件在发送之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.11 tliMSend_m_MC

标记	void tliMSend_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送消息经由的端口。
	msgValue	待编码和发送的值。
	addresses	在 SUT 内目的地地址。
	encoderFailure	在编码时可能会失败的消息。
	msg	已编码的消息。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录多播发送操作。该事件在发送之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.12 tliMSend_c

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

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送消息经由的端口。
	`msgValue`	待编码和发送的值。
	`to`	应收到消息的部件。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 CH 调用以记录单播发送操作。该事件在发送之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.13 tliMSend_c_BC

标记	`void tliMSend_c_BC(in TString am, in TInteger ts,`
	`in TString src, in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port, in Value msgValue,`
	`in TriStatusType transmissionFailure)`	

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送消息经由的端口。
	`msgValue`	待编码和发送的值。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 CH 调用以记录广播发送操作。该事件在发送之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.14 tliMSend_c_MC

标记	`void tliMSend_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送消息经由的端口。
	`msgValue`	待编码和发送的值。
	`toList`	应收到消息的部件。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 CH 调用以记录多播发送操作。该事件在发送之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.15 tliMDetected_m

标记	`void tliMDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriMessageType msg, in TriAddressType address)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收消息经由的端口。
	`msg`	接收到的编码消息。
	`address`	SUT 内的源地址。
返回值	`void`	
约束条件	应由 SA 调用以记录排队的消息。该事件在消息排队之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.16 tliMDetected_c

标记	void tliMDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType from)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收消息经由的端口。
	msgValue	接收到的消息。
	from	发送消息的部件。
返回值	void	
约束条件	应由 CH 调用以记录排队的消息。该事件在消息排队之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.17 tliMMismatch_m

标记	void tliMMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收消息经由的端口。
	msgValue	按照模板检测的消息。
	msgTmpl	用于检测消息匹配的模板。
	diffs	消息和模板之间的不同/失配。
	address	SUT 内的源地址。
	addressTmpl	SUT 内期望的源地址。
返回值	void	
约束条件	应由 TE 调用以记录模板的失配。该事件在检测模板是否匹配之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.18 tliMMismatch_c

标记	`void tliMMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收消息经由的端口。
	`msgValue`	按照模板检测的消息。
	`msgTmpl`	用于检测消息匹配的模板。
	`diffs`	消息和模板之间的不同/失配。
	`from`	发送消息的部件。
	`fromTmpl`	期望的发送方部件。
返回值	`void`	
约束条件	应由 TE 调用以记录模板的失配。该事件在检测模板是否匹配之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.19 tliMReceive_m

标记	`void tliMReceive_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收消息经由的端口。
	`msgValue`	按照模板检测的消息。
	`msgTmpl`	用于检测消息匹配的模板。
	`address`	SUT 内的源地址。
	`addressTmpl`	SUT 内期望的源地址。
返回值	`void`	
约束条件	应由 TE 调用以记录消息的接收。该事件在检测模板是否匹配之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.20 tliMReceive_c

标记	`void tliMReceive_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收消息经由的端口。
	`msgValue`	按照模板检测的消息。
	`msgTmpl`	用于检测消息匹配的模板。
	`from`	发送消息的部件。
	`fromTmpl`	期望的发送方部件。
返回值	`void`	
约束条件	应由 TE 调用以记录消息的接收。该事件在检测模板是否匹配之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.21 tliPrCall_m

标记	`void tliPrCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发起调用的端口。
	`signature`	调用操作的标记。
	`parsValue`	调用操作的参数。
	`address`	在 SUT 内目的地地址。
	`encoderFailure`	在编码时可能会失败的消息。
	`pars`	已编码的参数。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 SA 调用以记录单播调用操作。该事件在调用执行之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.22 tliPrCall_m_BC

标记	void tliPrCall_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发起调用的端口。
	signature	调用操作的标记。
	parsValue	调用操作的参数。
	encoderFailure	在编码时可能会失败的消息。
	pars	已编码的参数。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录广播调用操作。该事件在调用执行之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.23 tliPrCall_m_MC

标记	void tliPrCall_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发起调用的端口。
	signature	调用操作的标记。
	parsValue	调用操作的参数。
	addresses	在 SUT 内目的地地址。
	encoderFailure	在编码时可能会失败的消息。
	pars	已编码的参数。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录多播调用操作。该事件在调用执行之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.24 tliPrCall_c

标记	void tliPrCall_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发起调用的端口。
	signature	调用操作的标记。
	parsValue	调用操作的参数。
	to	应收到消息的部件。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录单播调用操作。该事件在调用执行之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.25 tliPrCall_c_BC

标记	void tliPrCall_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发起调用的端口。
	signature	调用操作的标记。
	parsValue	调用操作的参数。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录广播调用操作。该事件在调用执行之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.26 tliPrCall_c_MC

标记	`void tliPrCall_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发起调用的端口。
	`signature`	调用操作的标记。
	`parsValue`	调用操作的参数。
	`toList`	应收到消息的部件。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 CH 调用以记录多播调用操作。该事件在调用执行之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.27 tliPrGetCallDetected_m

标记	`void tliPrGetCallDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterListType pars, in TriAddressType address)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	调用被接收的端口。
	`signature`	检测到调用的标记。
	`pars`	检测到调用的已编码参数。
	`address`	在 SUT 内目的地地址。
返回值	`void`	
约束条件	应由 SA 调用以记录 getcall 排队操作。该事件在调用排队之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.28 tliPrGetCallDetected_c

标记	void tliPrGetCallDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType from)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	调用被接收的端口。
	signature	调用操作的标记。
	parsValue	检测到调用的已编码参数。
	from	调用操作的部件。
返回值	void	
约束条件	应由 CH 调用以记录 getcall 排队操作。该事件在调用排队之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.29 tliPrGetCallMismatch_m

标记	void tliPrGetCallMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	调用被接收的端口。
	signature	检测到调用的标记。
	parsValue	检测到调用的参数。
	parsTmpl	用于检测参数匹配的模板。
	diffs	调用和模板之间的不同/失配。
	address	SUT 内的源地址。
	addressTmpl	SUT 内期望的源地址。
返回值	void	
约束条件	应由 TE 调用以记录 getcall 的失配。该事件在根据模板检测出 getcall 之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.30 tliPrGetCallMismatch_c

标记	`void tliPrGetCallMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	调用被接收的端口。
	`signature`	检测到调用的标记。
	`parsValue`	检测到调用的参数。
	`parsTmpl`	用于检测参数匹配的模板。
	`diffs`	消息和模板之间的不同/失配。
	`from`	调用操作的部件。
	`fromTmpl`	期望的调用部件。
返回值	`void`	
约束条件	应由 TE 调用以记录 getcall 的失配。该事件在根据模板检测出 getcall 之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.31 tliPrGetCall_m

标记	`void tliPrGetCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	调用被接收的端口。
	`signature`	检测到调用的标记。
	`parsValue`	检测到调用的参数。
	`parsTmpl`	用于检测参数匹配的模板。
	`address`	SUT 内的源地址。
	`addressTmpl`	SUT 内期望的源地址。
返回值	`void`	
约束条件	应由 TE 调用以记录调用的获得。该事件在 getcall 与模板匹配之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.32 tliPrGetCall_c

标记	`void tliPrGetCall_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	调用被接收的端口。
	`signature`	检测到调用的标记。
	`parsValue`	检测到调用的参数。
	`parsTmpl`	用于检测参数匹配的模板。
	`from`	调用操作的部件。
	`fromTmpl`	期望的调用部件。
返回值	`void`	
约束条件	应由 TE 调用以记录调用的获得。该事件在 getcall 与模板匹配之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.33 tliPrReply_m

标记	`void tliPrReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送应答经由的端口。
	`signature`	与应答相关的标记。
	`parsValue`	与应答相关的标记参数。
	`replValue`	待发送的应答。
	`address`	在 SUT 内目的地地址。
	`encoderFailure`	在编码时可能会失败的消息。
	`repl`	已编码的应答。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 SA 调用以记录单播应答操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.34 tliPrReply_m_BC

标记	void tliPrReply_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	待发送的应答。
	encoderFailure	在编码时可能会失败的消息。
	repl	已编码的应答。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录广播应答操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.35 tliPrReply_m_MC

标记	void tliPrReply_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	待发送的应答。
	addresses	在 SUT 内目的地地址。
	encoderFailure	在编码时可能会失败的消息。
	repl	已编码的应答。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录多播应答操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.36 tliPrReply_c

标记	void tliPrReply_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	待发送的应答。
	to	应接收到应答的部件。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录单播应答操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.37 tliPrReply_c_BC

标记	void tliPrReply_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	待发送的应答。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录广播应答操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.38 tliPrReply_c_MC

标记	void tliPrReply_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	待发送的应答。
	toList	应接收到应答的部件。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录多播应答操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.39 tliPrGetReplyDetected_m

标记	void tliPrGetReplyDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterType repl, in TriAddressType address)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收应答经由的端口。
	signature	与应答相关的标记。
	repl	接收到的编码应答。
	address	SUT 内的源地址。
返回值	void	
约束条件	应由 SA 调用以记录 getreply 排队操作。该事件在 getreply 排队之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.40 tliPrGetReplyDetected_c

标记	`void tliPrGetReplyDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value replValue, in TriComponentIdType from)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收应答经由的端口。
	`signature`	与应答相关的标记。
	`replValue`	接收到的应答。
	`from`	发送应答的部件。
返回值	`void`	
约束条件	应由 CH 调用以记录 getreply 排队操作。该事件在 getreply 排队之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.41 tliPrGetReplyMismatch_m

标记	`void tliPrGetReplyMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收应答经由的端口。
	`signature`	与应答相关的标记。
	`parsValue`	与应答相关的标记参数。
	`replValue`	接收到的应答。
	`replyTmpl`	用于检测应答是否匹配的模板。
	`diffs`	应答和模板之间的不同/失配。
	`address`	SUT 内的源地址。
	`addressTmpl`	SUT 内期望的源地址。
返回值	`void`	
约束条件	应由 TE 调用以记录 getreply 操作的失配。该事件在根据模板检测 getreply 之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.42 tliPrGetReplyMismatch_c

标记	void tliPrGetReplyMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	接收到的应答。
	replyTmpl	用于检测应答是否匹配的模板。
	diffs	应答和模板之间的不同/失配。
	from	发送应答的部件。
	fromTmpl	期望的应答部件。
返回值	void	
约束条件	应由 TE 调用以记录 getreply 操作的失配。该事件在根据模板检测 getreply 之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.43 tliPrGetReply_m

标记	void tliPrGetReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收应答经由的端口。
	signature	与应答相关的标记。
	parsValue	与应答相关的标记参数。
	replValue	接收到的应答。
	replyTmpl	用于检测应答是否匹配的模板。
	address	SUT 内的源地址。
	addressTmpl	SUT 内期望的源地址。
返回值	void	
约束条件	应由 TE 调用以记录应答的获得。该事件在根据模板检测 getreply 之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.44 tliPrGetReply_c

标记	`void tliPrGetReply_c(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port, in TriSignatureIdType signature,`
	`in TciParameterListType parsValue, in Value replValue,`
	`in TciValueTemplate replyTmpl,`
	`in TriComponentIdType from,`
	`in TciNonValueTemplate fromTmpl)`	

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收应答经由的端口。
	`signature`	与应答相关的标记。
	`parsValue`	与应答相关的标记参数。
	`replValue`	接收到的应答。
	`replyTmpl`	用于检测应答是否匹配的模板。
	`from`	发送应答的部件。
	`fromTmpl`	期望的应答部件。
返回值	`void`	
约束条件	应由 TE 调用以记录应答的获得。该事件在根据模板检测 getreply 之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.45 tliPrRaise_m

标记	`void tliPrRaise_m(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port, in TriSignatureIdType signature,`
	`in TciParameterListType parsValue, in Value excValue,`
	`in TriAddressType address,`
	`in TriStatusType encoderFailure,`
	`in TriExceptionType exc,`
	`in TriStatusType transmissionFailure)`	

输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	发送例外情况经由的端口。
	`signature`	与例外情况有关的标记。
	`parsValue`	与例外情况有关的标记参数。
	`excValue`	待发送的例外情况。
	`address`	在 SUT 内目的地地址。
	`encoderFailure`	在编码时可能会失败的消息。
	`exc`	已编码的例外情况。
	`transmissionFailure`	在发送时可能会失败的消息。
返回值	`void`	
约束条件	应由 SA 调用以记录引起单播操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.46 tliPrRaise_m_BC

标记	void tliPrRaise_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	待发送的例外情况。
	encoderFailure	在编码时可能会失败的消息。
	exc	已编码的例外情况。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录引起广播操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.47 tliPrRaise_m_MC

标记	void tliPrRaise_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	待发送的例外情况。
	addresses	在 SUT 内目的地地址。
	encoderFailure	在编码时可能会失败的消息。
	exc	已编码的例外情况。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 SA 调用以记录引起多播操作。该事件在执行应答之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.48 tliPrRaise_c

标记	void tliPrRaise_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	待发送的例外情况。
	to	应接收到应答的部件。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录引起单播操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.49 tliPrRaise_c_BC

标记	void tliPrRaise_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	待发送的例外情况。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录引起广播操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.50 tliPrRaise_c_MC

标记	void tliPrRaise_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	发送例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	待发送的例外情况。
	toList	应接收应答的部件。
	transmissionFailure	在发送时可能会失败的消息。
返回值	void	
约束条件	应由 CH 调用以记录引起多播操作。该事件在执行应答之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.51 tliPrCatchDetected_m

标记	void tliPrCatchDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收例外情况经由的端口。
	signature	与例外情况有关的标记。
	exc	捕捉到的例外情况。
	address	SUT 内的源地址。
返回值	void	
约束条件	应由 SA 调用以记录捕捉排队操作。该事件在捕捉排队之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.52 tliPrCatchDetected_c

标记	`void tliPrCatchDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value excValue, in TriAddressType address)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收例外情况经由的端口。
	`signature`	与例外情况有关的标记。
	`excValue`	捕捉到的例外情况。
	`address`	SUT 内的源地址。
返回值	`void`	
约束条件	应由 CH 调用以记录捕捉排队操作。该事件在捕捉排队之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.53 tliPrCatchMismatch_m

标记	`void tliPrCatchMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收例外情况经由的端口。
	`signature`	与例外情况有关的标记。
	`parsValue`	与例外情况有关的标记参数。
	`excValue`	接收到的例外情况。
	`excTmpl`	用于检测例外情况匹配的模板。
	`diffs`	例外情况和模板之间的不同/失配。
	`address`	SUT 内的源地址。
	`addressTmpl`	SUT 内期望的源地址。
返回值	`void`	
约束条件	应由 TE 调用以记录捕捉操作的失配。该事件在根据模板检测捕捉之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.54 tliPrCatchMismatch_c

标记	void tliPrCatchMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	接收到的例外情况。
	excTmpl	用于检测例外情况匹配的模板。
	diffs	例外情况和模板之间的不同/失配。
	from	发送应答的部件。
	fromTmpl	期望的应答部件。
返回值	void	
约束条件	应由 TE 调用以记录捕捉操作的失配。该事件在根据模板检测捕捉之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.55 tliPrCatch_m

标记	void tliPrCatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收例外情况经由的端口。
	signature	与例外情况有关的标记。
	parsValue	与例外情况有关的标记参数。
	excValue	接收到的例外情况。
	excTmpl	用于检测例外情况匹配的模板。
	address	SUT 内的源地址。
	addressTmpl	SUT 内期望的源地址。
返回值	void	
约束条件	应由 SA 调用以记录例外情况的捕捉。该事件在根据模板检测捕捉之后发生。该事件用于记录与 SUT 的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.56 tliPrCatch_c

标记	`void tliPrCatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收例外情况经由的端口。
	`signature`	与例外情况有关的标记。
	`parsValue`	与例外情况有关的标记参数。
	`excValue`	接收到的例外情况。
	`excTmpl`	用于检测例外情况匹配的模板。
	`from`	发送应答的部件。
	`fromTmpl`	期望的应答部件。
返回值	`void`	
约束条件	应由 CH 调用以记录例外情况的捕捉。该事件在根据模板检测捕捉之后发生。该事件用于记录部件间的通信。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.57 tliPrCatchTimeoutDetected

标记	`void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`port`	接收例外情况经由的端口。
	`signature`	与例外情况有关的标记。
返回值	`void`	
约束条件	应由 PA 调用以记录检测出捕捉超时。该事件在超时排队之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.58 tliPrCatchTimeout

标记	void tliPrCatchTimeout(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	接收例外情况经由的端口。
	signature	与例外情况有关的标记。
返回值	void	
约束条件	应由 TE 调用以记录超时的捕捉。该事件在实施捕捉超时之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.59 tliCCreate

标记	void tliCCreate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TString name)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	被创建的部件。
	name	被创建的部件名称。
返回值	void	
约束条件	应由 TE 调用以记录创建部件操作。该事件在部件创建之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.60 tliCStart

标记	void tliCStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciBehaviourIdType beh, in TciParameterListType pars)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	被启动的部件。
	beh	在部件上被启动的行为。
	pars	被启动的行为参数。
返回值	void	
约束条件	应由 TE 调用以记录启动部件操作。该事件在部件启动之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.61 tliCRunning

标记	void tliCRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	检测到正在运行的部件。
	status	部件的状态。
返回值	void	
约束条件	应由 TE 调用以记录运行部件操作。该事件在部件运行之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.62 tliCAlive

标记	void tliCAlive(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	检测到正在运行的部件。
	status	部件的状态。
返回值	void	
约束条件	应由 TE 调用以记录激活的部件操作。该事件在部件激活之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.63 tliCStop

标记	void tliCStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	被停止的部件。
返回值	void	
约束条件	应由 TE 调用以记录停止部件操作。该事件在部件停止之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.64 tliCKill

标记	void tliCKill(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	comp	被停止的部件。
返回值	void	
约束条件	应由 TE 调用以记录取消部件操作。该事件在部件取消之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.65 tliCDoneMismatch

标记	void tliCDoneMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	compTmpl	用于检测完成匹配的模板。
返回值	void	
约束条件	应由 TE 调用以记录完成部件操作的失配。该事件在按照模板检测到完成之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.66 tliCDone

标记	void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	compTmpl	用于检测完成匹配的模板。
返回值	void	
约束条件	应由 TE 调用以记录完成部件操作。该事件在完成操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.67 tliCKilledMismatch

标记	`void tliCKilledMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`compTmpl`	用于检测完成匹配的模板。
返回值	`void`	
约束条件	应由 TE 调用以记录取消部件操作的失配。该事件在按照模板检测到取消之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.68 tliCKilled

标记	`void tliCKilled(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`compTmpl`	用于检测完成匹配的模板。
返回值	`void`	
约束条件	应由 TE 调用以记录取消部件操作。该事件在取消操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.69 tliCTerminated

标记	`void tliCTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`verdict`	部件的判定。
返回值	`void`	
约束条件	应由 TE 调用以记录部件的终止。该事件在部件终止之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.70 tliPConnect

标记	void tliPConnect(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	c1	被连接的第一个端口的部件。
	port1	被连接的第一个端口。
	c2	被连接的第二个端口的部件。
	port2	被连接的第二个端口。
返回值	void	
约束条件	应由 CH 调用以记录连接操作。该事件在连接操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.71 tliPDisconnect

标记	void tliPDisconnect(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	c1	被断开连接的第一个端口的部件。
	port1	被断开连接的第一个端口。
	c2	被断开连接的第二个端口的部件。
	port2	被断开连接的第二个端口。
返回值	void	
约束条件	应由 CH 调用以记录断开连接操作。该事件在断开连接操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.72 tliPMap

标记	void tliPMap(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	c1	被映射的第一个端口的部件。
	port1	被映射的第一个端口。
	c2	被映射的第二个端口的部件。
	port2	被映射的第二个端口。
返回值	void	
约束条件	应由 SA 调用以记录映射操作。该事件在映射操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.73 tliPUnmap

标记	void tliPUnmap(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	c1	被取消映射的第一个端口的部件。
	port1	被取消映射的第一个端口。
	c2	被取消映射的第二个端口的部件。
	port2	被取消映射的第二个端口。
返回值	void	
约束条件	应由 SA 调用以记录取消映射操作。该事件在取消映射操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.74 tliPClear

标记	void tliPClear(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	要被清除的端口。
返回值	void	
约束条件	应由 TE 调用以记录端口清除操作。该事件在端口清除操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.75 tliPStart

标记	void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	要被启动的端口。
返回值	void	
约束条件	应由 TE 调用以记录端口启动操作。该事件在端口启动操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.76 tliPStop

标记	void tliPStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	要被停止的端口。
返回值	void	
约束条件	应由 TE 调用以记录端口停止操作。该事件在端口停止操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.77 tliPHalt

标记	void tliPHalt(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	port	要被停止的端口。
返回值	void	
约束条件	应由 TE 调用以记录端口暂停操作。该事件在端口暂停操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.78 tliEncode

标记	void tliEncode(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType encoderFailure, in TriMessageType msg, in TString codec)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	value	要被编码的值。
	encoderFailure	在编码时可能会失败的消息。
	msg	已编码的值。
	codec	使用的编码器。
返回值	void	
约束条件	应由 CD 调用以记录编码操作。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.79 tliDecode

标记	void tliDecode(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType decoderFailure, in TriMessageType msg, in TString codec)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	msg	要被解码的值。
	decoderFailure	在解码时可能出错的消息。
	value	已解码的值。
	codec	使用的解码器。
返回值	void	
约束条件	应由 CD 调用以记录解码操作。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.80 tliTTimeoutDetected

标记	void tliTTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	timer	定时器超时。
返回值	void	
约束条件	应由 PA 调用以记录检测到超时。该事件在超时排队之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.81 tliTTimeoutMismatch

标记	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timerTmpl`	定时器模板失配。
返回值	`void`	
约束条件	应由 TE 调用以记录超时失配。该事件在超时匹配出错之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.82 tliTTimeoutMismatch

标记	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timerTmpl`	定时器模板匹配。
返回值	`void`	
约束条件	应由 TE 调用以记录超时匹配。该事件在超时匹配之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.83 tliTStart

标记	`void tliTStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timer`	启动定时器。
	`dur`	定时器时间间隔。
返回值	`void`	
约束条件	应由 PA 调用以记录定时器的启动。该事件在定时器启动操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.84 tliTStop

标记	`void tliTStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timer`	停止定时器。
返回值	`void`	
约束条件	应由 PA 调用以记录定时器的停止。该事件在定时器停止操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.85 tliTRead

标记	`void tliTRead(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType elapsed)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timer`	启动定时器。
	`elapsed`	定时器使用的时间。
返回值	`void`	
约束条件	应由 PA 调用以记录定时器的读取。该事件在定时器读取操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.86 tliTRunning

标记	`void tliTRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`timer`	检测到正在运行的定时器。
	`status`	部件的状态。
返回值	`void`	
约束条件	应由 PA 调用以记录运行定时器的操作。该事件在运行定时器操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.87 tliSEnter

标记	`void tliSEnter(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType parsValue, in TString kind)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`name`	范围的名称。
	`parsValue`	范围的参数。
	`kind`	范围的种类。
返回值	`void`	
约束条件	应由 TE 调用以记录范围的输入。该事件在输入范围之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.88 tliSLeave

标记	`void tliSLeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value val, in TString kind)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`name`	范围的名称。
	`val`	范围的返回值。
	`kind`	范围的种类。
返回值	`void`	
约束条件	应由 TE 调用以记录范围的离开。该事件在离开范围之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.89 tliVar

标记	`void tliVar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value varValue)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`name`	变量的名称。
	`varValue`	变量的新值。
返回值	`void`	
约束条件	应由 TE 调用以记录变量值的修改。该事件在值已经发生改变之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.90 tliModulePar

标记	`void tliModulePar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value parValue)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`name`	模块参数的名称。
	`parValue`	模块参数的值。
返回值	`void`	
约束条件	应由 TE 调用以记录模块参数的值。该事件在接入到模块参数的值之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.91 tliGetVerdict

标记	`void tliGetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`verdict`	本地判定的现值。
返回值	`void`	
约束条件	应由 TE 调用以记录 getverdict 操作。该事件在 getverdict 操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.92 tliSetVerdict

标记	`void tliSetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
输入参数	`am`	额外的消息。
	`ts`	事件产生的时间。
	`src`	测试规范的源文件。
	`line`	执行请求的行号。
	`c`	产生该事件的部件。
	`verdict`	设置为本地判定的值。
返回值	`void`	
约束条件	应由 TE 调用以记录 setverdict 操作。该事件在 setverdict 操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.93 tliLog

标记	void tliLog(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciValueList log)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	log	要被记录的串。
返回值	void	
约束条件	应由 TM 调用以记录 TTCN-3 声明日志。该事件在 TTCN-3 记录操作之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.94 tliAEnter

标记	void tliAEnter(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录输入 alt。该事件在已经输入 alt 之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.95 tliALeave

标记	void tliALeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录 alt 的离开。该事件在 alt 已经离开之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.96 tliANomatch

标记	void tliANomatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录 alt 的没有匹配。该事件在 alt 没有匹配之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.97 tliARepeat

标记	void tliARepeat(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录 alt 的重复。该事件在正重复 alt 之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.98 tliADefaults

标记	void tliADefaults(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录输入缺省部分。该事件在已经输入缺省部分之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.99 tliAActivate

标记	void tliAActivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType pars, in Value ref)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	name	缺省的名称。
	pars	缺省的参数。
	ref	结果的缺省参考。
返回值	void	
约束条件	应由 TE 调用以记录缺省的激活。该事件在缺省激活之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.100 tliADeactivate

标记	void tliADeactivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value ref)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
	ref	结果的缺省参考。
返回值	void	
约束条件	应由 TE 调用以记录缺省的去激活。该事件在缺省去激活之后发生。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

7.3.4.1.101 tliAWait

标记	void tliAWait(in string am, in long ts, in string src, in long line, in TriComponentId c)	
输入参数	am	额外的消息。
	ts	事件产生的时间。
	src	测试规范的源文件。
	line	执行请求的行号。
	c	产生该事件的部件。
返回值	void	
约束条件	应由 TE 调用以记录部件等待新的快照事件。	
效果	TL 给出了在该操作的参数中提供给用户的所有信息，如何做到这一点不在本建议书的范围内。	

8 Java语言映射

8.1 引言

本节介绍了 TCI Java 语言映射。出于效率的原因，要专门引进语言映射，而不是使用 OMG IDL 到 Java 语言。

用于 TTCN-3 控制接口的 Java 语言映射规定了第 7 节中描述的 IDL 定义如何被映射到 Java 语言上。语言映射与所使用的 Java 版本无关，因为只使用了基本 Java 语言构建。

8.2 名称和范围

8.2.1 名称

虽然在 IDL 定义中使用的标识符和 Java 语言之间没有冲突，有些命名翻译规则还是适用于 IDL 标识符。

Java 接口或类别标识符忽略在 IDL 定义中使用的拖尾类型。

举例： IDL 类型 `TciTestCaseIdType` 映射到 Java 中的 `TciTestCaseId`。

结果的映射与标准的 Java 编码惯例一致。

8.2.2 范围

IDL 模块 **`tciInterface`** 映射到 Java 包 `org.etsi.ttcn3.tci`。在该模块内的所有 IDL 类型声明映射到该包内的 Java 类别或接口声明。

类型映射

基本类型映射

表 2 概述了初始类型 TBoolean、TFloat、TInteger、TString 和 TStringSeq 如何映射到 Java 类型。

表 2/Z.145－基本类型映射

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

初始类型 TObjId 在 ObjidValue 接口相应的部分中规定。

boolean

IDL **`TBoolean`** 类型映射到 java 基本类型 `boolean`。

float

IDL **`TFloat`** 类型映射到 java 基本类型 `float`。

char

IDL **`TChar`** 类型映射到 java 基本类型 `char`。

int

IDL **`TInteger`** 类型映射到 java 基本类型 `int`。

String

IDL **TString** 类型映射到 java.lang.String 类别，在该串中没有范围检查或字符限制。在 TTCN-3 中规定的所有可能的串能被转换成 java.lang.String。

String[]

IDL **TStringSeq** 类型映射到 java.lang.String 类别的阵列中。

通用字符

IDL **TUniversalChar** 类型映射到 java 基本类型 int。整数使用在 6.2/X.290 中规定的标准格式。

结构类型映射

TCI IDL 描述规定用户定义类型为初始类型。在 Java 语言映射中，这些类型被映射到 Java 接口。这些接口规定对象执行该接口可获得的方法和属性。

8.2.2.1 TciParameterType

TciParameterType 映射到下列接口：

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

方法：

- getParameterName() 返回参数名称，正如TTCN-3规范中规定的；
- setParameterName(String name) 把该TciParameter 参数的名称设置为name；
- getParameterPassingMode() 返回该参数的参数通过模式；
- setParameterPassingMode(TciParameterPassingMode mode) 把该TriParameter 参数的模式设置为mode；
- getParameter() 返回该TciParameter的值参数，或如果该参数包括特别值null，则返回null对象；
- setParameter(Value parameter) 把该 TciParameter 的值参数设置为 parameter。

  如果特别值null被置为指示该参数没有值，则应通过Java null为参数。

8.2.2.2 TciParameterPassingModeType

TciParameterPassingModeType 映射到下列接口：

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

常数：

- TCI_IN　　应被用于指示TciParameter是一个in参数；
- TCI_INOUT　　应被用于指示TciParameter是一个inout参数；
- TCI_OUT　　应被用于指示TciParameter是一个out参数。

8.2.2.3 TciParameterListType

TciParameterListType 映射到下列接口：

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

方法：

- size()　　返回该列表中的参数数量；
- isEmpty()　　如果该列表不包括参数，则返回true；
- getParameters()　　返回该列表中参数上的Enumeration。enumeration提供的参数顺序和他们在列表中出现的顺序相同；
- get(int index)　　在规定的位置返回TciParameter；
- clear()　　从该TciParameterList中去除所有的参数；
- add(TciParameter parameter)　　把parameter增加到该TciParameterList的末端；
- setParameter(TciParameter[] parameters)　　用parameters 填充该TciParameterList。

8.2.2.4 TciTypeClassType

TciTypeClassType 映射到下列接口：

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

8.2.2.5 TciTestComponentKindType

TciTestComponentKindType 映射到下列接口：

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

8.2.2.6 TciBehaviourIdType

TciBehaviourIdType 映射到下列接口：

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

8.2.2.7 TciTestCaseIdType

TciTestCaseIdType 映射到下列接口：

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

8.2.2.8 TciModuleIdType

TciModuleIdType 映射到下列接口：

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

8.2.2.9 TciModuleParameterIdType

TciModuleParameterIdType 映射到下列接口：

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

8.2.2.10 TciModuleParameterListType

TciModuleParameterListType 映射到下列接口：

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

方法：

- `size()` 返回该列表中的模块参数数量；
- `isEmpty()` 如果该列表不包括模块，则返回`true`；
- `getModuleParameters()` 返回该列表中模块参数上的`Enumeration`。`enumeration`提供的模块参数顺序和他们在列表中出现的顺序相同；
- `get(int index)` 在规定的位置返回`TciModuleParameter`。

8.2.2.11 TciModuleParameterType

TciModuleParameterType 映射到下列接口：

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

方法：

- `getModuleParameterName()` 返回模块参数名称，正如TTCN-3规范中规定的；
- `getDefaultValue()` 返回该`TciModuleParameter`的缺省值模块参数，或如果该模块参数包括特别值`null`，则返回`null`对象。

8.2.2.12 TciParameterTypeListType

TciParameterTypeListType 映射到下列接口：

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

方法：

- `size()` 返回该列表中的参数类型数量；
- `isEmpty()` 如果该列表不包括参数类型，则返回`true`；
- `getParameterTypes()` 返回该列表中参数类型上的`Enumeration`。`enumeration`提供的参数类型顺序和他们在列表中出现的顺序相同；
- `get(int index)` 在规定的位置返回`TciParameterType`。

8.2.2.13 TciModuleIdListType

TciModuleIdListType 映射到下列接口：

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

方法：

- `size()` 返回该列表中的模块数量；
- `isEmpty()` 如果该列表不包括模块，则返回`true`；
- `getImportedModules()` 返回该列表中模块上的`Enumeration`。`enumeration`提供的模块顺序和他们在列表中出现的顺序相同；
- `get(int index)` 在规定的位置返回`TciModuleId`。

8.2.3 抽象类型映射

TTCN-3 数据类型使用抽象类型映射在 Java 中建模，正如本节规定的。类型接口仅规定在 TTCN-3 规定类型中用于重新获取的操作。使用类型接口不能构建 TTCN-3 类型。使用单一接口类型建模类型，提供识别类型的方法和重新获取给定类型值的方法。

8.2.3.1 Type

Type 映射到下列接口：

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

方法：

- `getDefiningModule()` 返回已在模块中规定类型的模块标识符。如果类型代表TTCN-3基本类型，则返回特别值`null`；
- `getName()` 返回类型名称，正如TTCN-3模块中规定的；
- `getTypeClass()` 返回相应类型的类型类别。值`TciTypeClassType`有下列常数之一：`ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OBJID`, `OCTETSTRING`,`RECORD`, `RECORD_OF`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`,`VERDICT`；
- `newInstance()` 返回给定类型刚刚创建的值。未规定创建值的初始值；
- `getTypeEncoding()` 返回类型编码属性，正如TTCN-3模块中规定的；
- `getTypeEncodingVariant()` 该操作返回值编码变量属性，正如TTCN-3中规定的，如果有的话。如果没有规定编码变量属性，则返回特别值`null`；
- `getTypeEncoding()` 返回类型扩展属性，正如TTCN-3模块中规定的。

8.2.4 抽象值映射

TTCN-3 值能从 TE 重新获得并使用 Value 接口构建。值映射接口由 Value 如基本接口那样分级构建。已经为不同的类型值规定了专门的接口。

8.2.4.1 Value

Value 映射到下列接口：

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

方法：

- `getType()` 返回规定值的类型；
- `notPresent()` 如果规定值为`omit`，则返回`true`；否则返回`false`；
- `getValueEncoding()` 该操作返回值编码变量属性，正如TTCN-3中规定的，如果有的话。如果没有规定编码变量属性，则返回特别值`null`；

- `getValueEncodingVariant()` 该操作返回值编码变量属性，正如TTCN-3中规定的，如果有的话。如果没有规定编码变量属性，则返回特别值`null`。

8.2.4.2 IntegerValue

`IntegerValue` 类型映射到下列接口：

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

方法：

- `setInteger(int value)` 把该IntegerValue置为整数值`value`；
- `getInteger()` 返回该IntegerValue代表的整数值。

8.2.4.3 FloatValue

`FloatValue` 类型映射到下列接口：

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

方法：

- `setFloat(float value)` 把该FloatValue置为浮点值`value`；
- `getFloat()` 返回该FloatValue代表的浮点值。

8.2.4.4 BooleanValue

`BooleanValue` 类型映射到下列接口：

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

方法：

- `setBoolean(boolean value)` 把该BooleanValue置为布尔值`value`；
- `getBoolean()` 返回该BooleanValue代表的布尔值。

8.2.4.5 ObjidValue

`ObjidValue` 映射到下列接口：

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

方法：

- `getObjid()` 返回该TTCN-3 `objid`的对象标识符值；
- `setObjid(TciObjId value)` 把该`ObjidValue`置为`value`。

8.2.4.6 TciObjId

TciObjId 映射到下列接口。TTCN-3 ObjectId 的初始 java 表示法包括一个 TciObjIdElement 的有序序列。.

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

方法：

- `size()` 返回TciObjIdElements 中该Object Id的大小；
- `setObjElement(TciObjIdElement[] objElements)` 把该ObjId置为objElements的列表；
- `getObjElement(int index)` 在index 位置处返回TciObjIdElement。

8.2.4.7 TciObjIdElement

TciObjIdElement 代表一个 TTCN-3 ObjId 值里的简单对象单元。可以使用不同的表示法例如 ASCII 表示法或整数法来设置。

TciObjIdElement 映射到下列接口：

```
package org.etsi.ttcn.tci;
public interface TciObjIdElement {
    public void     setElementAsAscii(String element) ;
    public void     setElementAsNumber(int element) ;
    public String   getElementAsAscii() ;
    public int      getElementAsNumber() ;
}
```

方法：

- `setElementAsAscii(String element)` 把该ObjIdElement的内部表示法置为串值element；
- `setElementAsNumber(int element)` 把该ObjIdElement的内部表示法置为整数值element；
- `getElementAsAscii()` 返回该ObjIdElement的内部表示法为串值；
- `getElementAsNumber()` 把该ObjIdElement的内部表示法置为整数值。

8.2.4.8 CharstringValue

CharstringValue 映射到下列接口：

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

方法：

- `getString()` 返回该TTCN-3 charstring的串值。空TTCN-3 charstring 的文本表示法为''，而其长度为0；
- `setString(String value)` 把该CharstringValue置为value；

- `getChar(int position)` 在position处返回TTCN-3字符串的字符值。Position 0表示TTCN-3字符串的第一个字符。position的有效值是0到长度-1；
- `setChar(int position, char value)` 把position 处的字符置为value。position的有效值是0到长度-1；
- `getLength()` 返回字符中该CharstringValue的长度，如果该CharstringValue的值为omit，则长度为0；
- `setLength(int len)` 把字符中该CharstringValue的长度置为len。

8.2.4.9 BitstringValue

`BitstringValue` 映射到下列接口：

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

方法：

- `getString()` 返回该BitstringValue的文本表示，正如TTCN-3中规定的；例如， 0101的文本表示为"0101"B。空的TTCN-3 bitstring的文本表示为""B，而长度为0；
- `setString(String value)` 根据value规定的文本表示来设置该BitstringValue的值。例如，如果在值中的文本表示为"0101"B，则该BitstringValue的值为0101；
- `getBit(int position)` 在该TTCN-3比特串的position返回值(0 | 1)。Position 0表示TTCN-3比特串的第一个比特。position的有效值是0到长度-1；
- `setBit(int position, int value)` 在position处把比特置为值(0 | 1)。Position 0表示该BitstringValue的第一个比特。position的有效值是0到长度-1；
- `getLength()` 返回比特中该BitstringValue的长度，如果该BitstringValue的值为omit，则长度为0；
- `setLength(int len)` 把比特中该BitstringValue的长度置为len。

8.2.4.10 OctetstringValue

`OctetstringValue` 映射到下列接口：

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

方法：

- `getString()` 返回该OctetstringValue的文本表示，正如TTCN-3中规定的；例如，0xCAFFEE的文本表示为"CAFFEE"O。空的TTCN-3 octetstring的文本表示为""O，而长度为0；
- `setString(String value)` 根据value规定的文本表示来设置该OctetstringValue的值。例如，如果在值中的文本表示为"CAFFEE"O，则该OctetstringValue的值为0xCAFFEE；
- `getOctet(int position)` 在该TTCN-3字节串的position返回值（0..255）。Position 0表示TTCN-3字节串的第一个字节。position的有效值是0到长度-1；
- `setOctet(int position, int value)` 在position处把字节置为值（0..255）。Position 0表示该字节串的第一个字节。position的有效值是0到长度-1；
- `getLength()` 返回字节中该OctetstringValue的长度，如果该OctetstringValue的值为omit，则长度为0；
- `setLength(int len)` 把字节中该OctetstringValue的长度置为len。

8.2.4.11 UniversalCharstringValue

UniversalCharstringValue 映射到下列接口：

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

方法：

- `getString()` 返回该UniversalCharstringValue的文本表示，正如TTCN-3中规定的；
- `setString(String value)` 根据value规定的文本表示来设置该UniversalCharstringValue的值；
- `getChar(int position)` 在该TTCN-3 universal charstring的position返回UniversalChar值。Position 0表示TTCN-3 universal charstring的第一个UniversalChar。position的有效值是0到长度-1；
- `setChar(int position, char value)` 在position处把UniversalChar置为value。Position的有效值是0到长度-1；
- `getLength()` 返回UniversalChars中该UniversalCharstringValue的长度，如果该UniversalCharstringValue的值为omit，则长度为0；
- `setLength(int len)` 把UniversalChars中该UniversalCharstringValue的长度置为len。

8.2.4.12 HexstringValue

HexstringValue 映射到下列接口：

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

方法：

- `getString()` 返回该HextstringValue的文本表示，正如TTCN-3中规定的；例如，0xAFFEE的文本表示为"AFFEE"H。空的TTCN-3hexstring的文本表示为""H，而长度为0；
- `setString(String value)` 根据value规定的文本表示来设置该HexstringValue的值。例如，如果在值中的文本表示为"AFFEE"H，则该HexstringValue的值为0xAFFEE；
- `getHex(int position)` 在该TTCN-3 hexstring的position返回值(0..15)。Position 0表示TTCN-3 hexstring的第一个hex digits。position的有效值是0到长度-1；
- `setHex(int position, int value)` 在position处把hex digit置为值(0..16)。Position 0表示hexstring的第一个字节。position的有效值是0到长度-1；
- `getLength()` 返回字节中该HexstringValue的长度，如果该HexstringValue的值为omit，则长度为0；
- `setLength(int len)` 把hex digits中该HexstringValue的长度置为len。

8.2.4.13 RecordValue

RecordValue 映射到下列接口：

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

方法：

- `getField(String fieldName)` 返回命名为fieldName 的字段值。返回值为常用抽象库类型Value，正如TTCN-3中规定的记录字段能有的任何类型。如果不能从记录中获得字段，应返回特别值null；
- `setField(String fieldName, Value value)` 把记录中命名为fieldName的字段置为value。对字段如何存储在记录中不必做出假设。内部执行时可能选择存储参考值到该值中或复制该值。假定该值应被复制是安全的。因此，应该假定value的后续修改不会反映在记录中；
- `getFieldNames()` 返回字段名称串的阵列，如果记录没有字段，则为特别值null。

8.2.4.14 RecordOfValue

RecordOfValue 映射到下列接口：

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

方法：

- `getField(String fieldName)` 如果position在0和长度-1之间，则返回在position处记录的值，否则返回特别值null。返回值为常用抽象库类型Value，正如TTCN-3中规定的记录字段能有的任何类型；
- `setField(int position, Value value)` 把position处的字段置为value。如果position大于(长度-1)，则记录应扩展到长度(position + 1)。应把length和position - 1的原始位置之间的单元记录置为OMIT。对字段如何存储在记录中不必做出假设。内部执行时可能选择存储参考值到该值中或复制该值。假定该值应被复制是安全的。因此，应该假定value的后续修改不会反映在记录中；
- `appendField(Value value)` 在record的末端添加值，即在位置length处。对字段如何存储在记录中不必做出假设。内部执行时可能选择存储参考值到该值中或复制该值。假定该值应被复制是安全的。因此，应该假定value的后续修改不会反映在记录中；
- `getElementType()` 该操作应返回该记录的单元类型；
- `getLength()` 返回值的记录的实际长度，如果值的记录为OMIT，则返回0；
- `setLength(int len)` 把该记录的长度置为len。如果len大于初始长度，则新创建的单元值为OMIT。如果len小于或等于初始长度，则忽略该操作。

8.2.4.15 UnionValue

UnionValue 映射到下列接口：

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

方法：

- `getVariant(String variantName)` 如果variantName等于getPresentVariantName的结果，则返回TTCN-3集合variantName的值，否则返回特别值null。variantName表示集合变量的名称，正如TTCN-3中规定的；
- `setVariant(String variantName, Value value)` 把集合的variantName置为value。如果variantName不是为该集合规定的，则忽略该操作。如果选择了另一个变量，应选择新的变量来替代；
- `getPresentVariantName()` 返回在该集合中值置为String的变量名称。如果没有选择变量，应返回特别值null；
- `getVariantNames()` 返回变量名称串的阵列，如果该集合没有字段，则为特别值null。如果UnionValue代表TTCN-3 anytype，即由getType()获得的类型类别为ANYTYPE，应返回所有预先规定和用户规定的TTCN-3 类型。

8.2.4.16 EnumeratedValue

EnumeratedValue 映射到下列接口：

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

方法：

- `getEnum()` 返回该EnumeratedValue的串标识符。该标识符等于TTCN-3规范中的标识符；
- `setEnum(String enumValue)` 把enum置为enumValue。如果对于这次枚举，该enumValue不是允许的值，应忽略该操作。

8.2.4.17 VerdictValue

VerdictValue 映射到下列接口：

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

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

方法：

- `getVerdict()` 对于该VerdictValue返回整数值。integer是下列常数之一：ERROR, FAIL, INCONC, NONE, PASS；
- `setVerdict(int verdict)` 把该VerdictValue置为verdict。注意到可以随时把VerdictValue置为以上提到的判定中的任一个。VerdictValue不执行任何判定计算，正如TTCN-3中规定的。例如，首先把VerdictValue置为ERROR，然后置为PASS是允许的。

8.2.4.18 AddressValue

AddressValue 映射到下列接口：

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

方法：

- getAddress()　　返回由该AddressValue代表的值；
- setAddress(Value value)　　把该AddressValue置为值value。

8.2.5 抽象记录类型映射

规定额外类型以方便值和模板之间的匹配记录。

8.2.5.1 TciValueTemplate

TciValueTemplate 映射到下列接口：

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

方法：

- isOmit()　　如果模板是omit，则返回true，否则返回false；
- isAny()　　如果模板是any，则返回true，否则返回false；
- isAnyOrOmit()　　如果模板是anyoromit，则返回true，否则返回false；
- getTemplateDef()　　该操作返回模板定义。

8.2.5.2 TciNonValueTemplate

TciNonValueTemplate 映射到下列接口：

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

方法：

- isAny()　　如果模板是any，则返回true，否则返回false；
- isAll()　　如果模板是all，则返回true，否则返回false；
- getTemplateDef()　　该操作返回模板定义。

8.2.5.3 TciValueList

TciValueList 映射到下列接口：

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

方法：

- `size()` 返回该列表中的值的数量；
- `isEmpty()` 如果列表中没有包含值，则返回`true`；
- `get(int index)` 在规定的位置返回`Value`。

8.2.5.4 TciValueDifference

`TciValueDifference` 映射到下列接口：

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

方法：

- `getValue()` 返回该`TciValueDifference`的值；
- `getTciValueTemplate()` 返回该`TciValueDifference`的模板；
- `getDescription()` 返回失配的描述。

8.2.5.5 TciValueDifferenceList

`TciValueDifferenceList` 映射到下列接口：

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

方法：

- `size()` 返回该列表中差别的数量；
- `isEmpty()` 如果该列表中不包含差别，则返回`true`；
- `get(int index)` 在规定的位置返回`TciValueDifference`。

8.3 常数

在该 Java 中，已经规定了语言映射常数。规定所有常数为 `public final static`，且可以从每个包的每个对象获得。在本节中规定的常数没有和 IDL 一起规定。他们由标注为初始类型的 TCI IDL 类型的规范产生。

下列常数能被用于确定 TTCN-3 参数的参数通过模式（也见 8.2.2.2）。

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

对于特别参数值 `null`，已编码参数值应被置为 Java `null`。

下列常数应用于值的处理（也见 8.2.2.4）。

- `org.etsi.ttcn.tci.TciTypeClass.ADDRESS`;
- `org.etsi.ttcn.tci.TciTypeClass.ANYTYPE`;
- `org.etsi.ttcn.tci.TciTypeClass.BITSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.BOOLEAN`;
- `org.etsi.ttcn.tci.TciTypeClass.CHARSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.COMPONENT`;

- org.etsi.ttcn.tci.TciTypeClass.ENUMERATED;
- org.etsi.ttcn.tci.TciTypeClass.FLOAT;
- org.etsi.ttcn.tci.TciTypeClass.HEXSTRING;
- org.etsi.ttcn.tci.TciTypeClass.INTEGER;
- org.etsi.ttcn.tci.TciTypeClass.OBJID;
- org.etsi.ttcn.tci.TciTypeClass.OCTETSTRING;
- org.etsi.ttcn.tci.TciTypeClass.RECORD;
- org.etsi.ttcn.tci.TciTypeClass.RECORD_OF;
- org.etsi.ttcn.tci.TciTypeClass.SET;
- org.etsi.ttcn.tci.TciTypeClass.SET_OF;
- org.etsi.ttcn.tci.TciTypeClass.UNION;
- org.etsi.ttcn.tci.TciTypeClass.UNIVERSAL_CHARSTRING;
- org.etsi.ttcn.tci.TciTypeClass.VERDICT。

下列常数应用于部件处理（也见 8.2.2.5）。

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

8.4 接口的映射

TCI IDL 定义规定了 4 个接口：**TCI-TM**、**TCI-CH**、**TCI-CD** 和 **TCI-TL** 接口。规定了该接口内不同方向的操作；即在测试管理和控制（TMC）上，某些操作仅能由可执行的 TTCN-3（TE）、系统适配器（SA）或平台适配器（PA）调用，而在 TE 上仅能由 TMC 调用其他操作。这体现在把 TCI IDL 接口分为两个子接口，每个的后缀为 Required 或 Provided。

表 3/Z.145－子接口

调用/被调用	TE	TM	CD	CH	SA	PA	TL
TE	–	提供的 TCI-TM	提供的 TCI-CD	提供的 TCI-CH	见 ITU-T Z.144 建议书 [1]	见 ITU-T Z.144 建议书 [1]	提供的 TCI-TL
TM	要求的 TCI-TM	–	–	–	–	–	提供的 TCI-TL
CD	要求的 TCI-CD	–	–	–	–	–	提供的 TCI-TL
CH	要求的 TCI-CH	–	–	–	–	–	提供的 TCI-TL
SA	见 ITU-T Z.144 建议书	–	–	–	–	–	提供的 TCI-TL
PA	见 ITU-T Z.144 建议书	–	–	–	–	–	提供的 TCI-TL
TL	–	–	–	–	–	–	–

在这些接口中规定的所有方法应如第 7 节中规定的那样执行。

8.4.1 TCI-TM接口

TCI-TM 接口分为两个子接口：提供的 TCI-TM 接口，规定从 TE 到 TM 的调用以及要求的 TCI-TM 接口，规定从 TM 到 TE 的调用。

8.4.1.1 提供的TCI-TM

提供的 TCI-TM 接口映射到下列接口：

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

8.4.1.2 要求的TCI-TM

要求的 TCI-TM 接口映射到下列接口：

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

8.4.2 TCI-CD接口

TCI-CD 接口分为两个子接口：提供的 TCI-CD 接口，规定从 TE 到 CD 的调用以及要求的 TCI-CD 接口，规定从 CD 到 TE 的调用。

8.4.2.1 提供的TCI-CD

提供的 TCI-CD 接口映射到下列接口：

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

8.4.2.2 要求的TCI-CD

要求的 TCI-CD 接口映射到下列接口：

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

8.4.3 TCI-CH接口

TCI-CH 接口分为两个子接口：提供的 TCI-CH 接口，规定从 TE 到 CH 的调用以及要求的 TCI-CH 接口，规定从 CH 到 TE 的调用。

8.4.3.1 提供的TCI-CH

提供的 TCI-CH 接口映射到下列接口：

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

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

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

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

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

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

    public void     tciStopTestComponentReq(TriComponentId comp) ;

    public void     tciConnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void     tciDisconnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void     tciTestComponentTerminatedReq(TriComponentId comp, VerdictValue verdict) ;

    public boolean  tciTestComponentRunningReq(TriComponentId comp) ;

    public TriComponentId   tciGetMTCReq() ;

    public void     tciMapReq(TriPortId fromPort, TriPortId toPort);
```

```
    public void     tciUnmapReq(TriPortId fromPort, TriPortId toPort);

    public void     tciExecuteTestCaseReq(TriComponentId testComponentId,
                                          TriPortIdList tsiPortList);

    public void     tciResetReq() ;

    public boolean  tciTestComponentDoneReq(TriComponentId testComponentId) ;

    public void     tciKillTestComponentReq(TriComponentId component)

    public boolean  tciTestComponentAliveReq(TriComponentId component)

    public boolean  tciTestComponentKilledReq(TriComponentId component)
}
```

8.4.3.2 要求的TCI-CH

要求的 TCI-CH 接口映射到下列接口：

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

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

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

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

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

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

    public void     tciStopTestComponent(TriComponentId comp) ;

    public void     tciConnect(TriPortId fromPort, TriPortId toPort) ;

    public void     tciDisconnect(TriPortId fromPort, TriPortId toPort) ;

    public void     tciTestComponentTerminated(TriComponentId comp, VerdictValue verdict);

    public boolean  tciTestComponentRunning(TriComponentId comp);

    public boolean  tciTestComponentDone(TriComponentId comp);

    public TriComponentId   tciGetMTC();

    public void     tciExecuteTestCase(TciTestCaseId TestCaseId, TriPortIdList tsiPortList);

    public void     tciReset();

    public void     tciMap(TriPortId fromPort, TriPortId toPort);

    public void     tciUnmap(TriPortId fromPort, TriPortId toPort);

    public void     tciKillTestComponent(TriComponentId component)

    public boolean  tciTestComponentAlive(TriComponentId component)

    public boolean  tciTestComponentKilled(TriComponentId component)
}
```

8.4.4 TCI-TL接口

TCI-TL 接口仅包括 1 个子接口：提供的 TCI-TL 接口，规定从 TE、TM、CH、CD、SA 和 PA 到 TL 的调用。

8.4.4.1 提供的TCI-TL

提供的 TCI-TL 接口映射到下列接口：

```
// TCI-TL
// TE,TM,CH,CD,SA,PA -> TL
package org.etsi.ttcn.tci;
public interface TciTLProvided {
    public void tliTcExecute(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TriParameterList pars, TriTimerDuration dur);
    public void tliTcStart(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TriParameterList pars, TriTimerDuration dur);
    public void tliTcStop(String am, int ts, String src, int line, TriComponentId c);
    public void tliTcStarted(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TriParameterList pars, TriTimerDuration dur);
    public void tliTcTerminated(String am, int ts, String src, int line, TriComponentId c,
            TciTestCaseId tcId, TriParameterList pars, VerdictValue outcome);
    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 port, Value msgValue, TriAddress address,
            TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue
            TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TriAddressList addresses,
            TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure);
    public void tliMSend_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TriComponentId to, TriStatus transmissionFailure);
    public void tliMSend_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TriStatus transmissionFailure);
    public void tliMSend_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TriComponentIdList toList,
            TriStatus transmissionFailure);
    public void tliMDetected_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriMessage msg, TriAddress address);
    public void tliMDetected_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TriComponentId from);
    public void tliMMismatch_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliMMismatch_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, 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 port, Value msgValue, TciValueTemplate msgTmpl,
            TriAddress address, TciValueTemplate addressTmpl);
    public void tliMReceive_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, Value msgValue, TciValueTemplate msgTmpl,
            TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrCall_m(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue,
            TriAddress address, TriStatus encoderFailure, TriParameterList pars,
            TriStatus transmissionFailure);
    public void tliPrCall_m_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue,
            TriStatus encoderFailure, TriParameterList pars,
            TriStatus transmissionFailure);
    public void tliPrCall_m_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue,
            TriAddressList addresses, TriStatus encoderFailure, TriParameterList pars,
            TriStatus transmissionFailure);
    public void tliPrCall_c(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue, TriComponentId to,
            TriStatus transmissionFailure);
    public void tliPrCall_c_BC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue,
            TriStatus transmissionFailure);
    public void tliPrCall_c_MC(String am, int ts, String src, int line, TriComponentId c,
            TriPortId port, TriSignatureId signature, TciParameterList parsValue,
            TriComponentIdList toList, TriStatus transmissionFailure);
    public void tliPrGetCallDetected_m(String am, int ts, String src, int line, TriComponentId c,
```

```
        TriPortId port, TriSignatureId signature, TriParameterList pars, TriAddress address);
public void tliPrGetCallDetected_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        TriComponentId from);
public void tliPrGetCallMismatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature,
        TciParameterList parsValue, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrGetCallMismatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature,
        TciParameterList parsValue, TciValueTemplate parsTmpl, TciValueDifferenceList diffs,
        TriComponentId from, TciNonValueTemplate fromTmpl);
public void tliPrGetCall_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature,
        TciParameterList parsValue, TciValueTemplate parsTmpl,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrGetCall_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature,
        TciParameterList parsValue, TciValueTemplate parsTmpl,
        TriComponentId from, TciNonValueTemplate fromTmpl);
public void tliPrReply_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriAddress address, TriStatus encoderFailure,
        TriParameter repl, TriStatus transmissionFailure);
public void tliPrReply_m_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriStatus encoderFailure,
        TriParameter repl, TriStatus transmissionFailure);
public void tliPrReply_m_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriAddressList addresses, TriStatus encoderFailure,
        TriParameter repl, TriStatus transmissionFailure);
public void tliPrReply_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriComponentId to, TriStatus transmissionFailure);
public void tliPrReply_c_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriStatus transmissionFailure);
public void tliPrReply_c_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value replValue,
        TriComponentIdList toList, TriStatus transmissionFailure);
public void tliPrGetReplyDetected_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TriParameter repl, TriAddress address);
public void tliPrGetReplyDetected_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, Value replValue, TriComponentId from);
public void tliPrGetReplyMismatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value replValue, TciValueTemplate replyTmpl,  TciValueDifferenceList diffs,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrGetReplyMismatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        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 port, TriSignatureId signature, TciParameterList parsValue,
        Value replValue, TciValueTemplate replyTmpl,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrGetReply_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value replValue, TciValueTemplate replyTmpl,
        TriComponentId from, TciNonValueTemplate fromTmpl);
public void tliPrRaise_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriAddress address, TriStatus encoderFailure, TriException exc,
        TriStatus transmissionFailure);
public void tliPrRaise_m_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriStatus encoderFailure, TriException exc, TriStatus transmissionFailure);
public void tliPrRaise_m_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriAddressList addresses, TriStatus encoderFailure, TriException exc,
        TriStatus transmissionFailure);
public void tliPrRaise_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
```

```
        TriComponentId to, TriStatus transmissionFailure);
    public void tliPrRaise_c_BC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriStatus transmissionFailure);
    public void tliPrRaise_c_MC(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriComponentIdList toList, TriStatus transmissionFailure);
    public void tliPrCatchDetected_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TriException exc, TriAddress address);
    public void tliPrCatchDetected_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, Value excValue, TriComponentId from);
    public void tliPrCatchMismatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,  TciValueDifferenceList diffs,
        TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrCatchMismatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        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 port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,
        TriAddress address, TciValueTemplate addressTmpl);
    public void tliPrCatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,
        TriComponentId from, TciNonValueTemplate fromTmpl);
    public void tliPrCatchTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature);
    public void tliPrCatchTimeout(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue);
    public void tliCCreate(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, String name);
    public void tliCStart(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TciBehaviourId name, TciParameterList parsValue);
    public void tliCRunning(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TBoolean status);
    public void tliCAlive(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TBoolean 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,
        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,
        TciNonValueTemplate compTmpl);
    public void tliCKilled(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate compTmpl);
    public void tliCTerminated(String am, int ts, String src, int line, TriComponentId c,
        VerdictValue verdict);
    public void tliPConnect(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPDisconnect(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPMap(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPUnmap(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
    public void tliPClear(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
    public void tliPStart(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
    public void tliPStop(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
    public void tliPHalt(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
    public void tliEncode(String am, int ts, String src, int line, TriComponentId c,
        Value val, TriStatus encoderFailure, TriMessage msg, String codec);
    public void tliDecode(String am, int ts, String src, int line, TriComponentId c,
        TriMessage msg, TriStatus 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,
        TciNonValueTemplate timerTmpl);
    public void tliTTimeout(String am, int ts, String src, int line, TriComponentId c,
        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);
    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, TBoolean status);
    public void tliSEnter(String am, int ts, String src, int line, TriComponentId c,
            String name, TciParameterList parsValue, String kind);
    public void tliSLeave(String am, int ts, String src, int line, TriComponentId c,
            String name, Value returnValue, String kind);
    public void tliVar(String am, int ts, String src, int line, TriComponentId c,
            String name, Value varValue);
    public void tliModulePar(String am, int ts, String src, int line, TriComponentId c,
            String name, Value parValue);
    public void tliGetVerdict(String am, int ts, String src, int line, TriComponentId c,
            VerdictValue verdict);
    public void tliSetVerdict(String am, int ts, String src, int line, TriComponentId c,
            VerdictValue verdict);
    public void tliLog(String am, int ts, String src, int line, TriComponentId c,
            TciValueList 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,
            String name, TciParameterList pars, 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);
}
```

8.5 选用参数

第 7.3 节规定保留值应被用于指示不存在选用参数。对于 Java 语言映射，Java `null` 值应被用于指示不存在选用值。例如，如果在 `tciReplyConnected` 操作中，应省略值参数，该操作调用为 `tciReplyConnected(sender, receiver, signature, parameterList, null)`。

8.6 TCI初始化

所有方法都是非静态的，即仅能在对象上调用操作。由于本建议书没有规定 TE、TM、CD 和 CH 具体的实施战略，因此 TE、TM、CD 或 CH 如何确定在相应的对象上处理的机制不在本建议书的范围内。

用具供货商应把方法提供给 TM、CD 和 CH 的开发商以把 TE、TM、CD 和 CH 注册到他们各自的通信方。

8.7 差错处理

所有操作均从 TM、CH 或 CD 调用，返回已成功完成。如果 TE 已经识别到错误的情况，则使用第 7.3.1.2.5 节（`tciError`）中规定的规程，把测试用例差错通知用户。如果在 TM、CH、CD 或 TL 中由 TE 调用的操作产生了差错，则应使用第 7.3.2.1.12 节（`tciErrorReq`）中规定的规程，把该差错情况通知 TE。

除了这种差错处理外，该 Java 语言映射没有规定额外的差错处理。特别是没有规定例外情况处理机制。

9 ANSI-C语言映射

9.1 引言

本节规定了用于第 7.2 节中规定的 TCI 数据和第 7.3 节中规定的 TCI 操作的 TRI ANSI-C 语言映射。

9.2 值的接口

TCI IDL接口	ANSI-C表示法	注释和说明
Type		
TciModuleIdType getDefiningModule()	`TciModuleIdType tciGetDefiningModule(TciType inst)`	
TString getName()	`String tciGetName(TciType inst)`	从 IDL 再使用的串类型（OMG 建议）
TciTypeClassType getTypeClass()	`TciTypeClassType tciGetTypeClass(TciType inst)`	
Value newInstance()	`TciValue tciNewInstance(TciType inst)`	
TString getTypeEncoding()	`String tciGetTypeEncoding(TciType inst)`	
TStringSeq getTypeExtension()	`String* getTypeExtension(TciType inst)`	
TString getTypeEncodingVariant()	`String tciGetTypeEncodingVariant(TciType inst)`	
Value		
TString getValueEncoding()	`String tciGetValueEncoding(TciValue inst)`	
TString getValueEncodingVariant()	`String tciGetValueEncodingVariant(TciValue inst)`	
Type getType()	`TciType tciGetType(TciValue inst)`	
TBoolean notPresent()	`Boolean tciNotPresent(TciValue inst)`	从 IDL 再使用的布尔类型（OMG 建议）
IntegerValue		
TInteger getInt()	`String tciGetIntAbs(TciValue inst)`	返回（10-base）整数绝对值作为一个 ASCII 串。
	`unsigned long int tciGetIntNumberOfDigits (TciValue inst)`	在一个整数值中返回数字的号码。
	`Boolean tciGetIntSign(TciValue inst)`	如果数为正，则返回 true，否则返回 false。
	`char tciGetIntDigit (TciValue inst, unsigned long int position)`	在位置'position'处返回数字的值，position 0 是最低有效位。
void setInt(in TInteger value)	`void tciSetIntAbs(TciValue inst, String value)`	通过一个 ASCII 串设置整数的（10-base） 绝对值。第一位不能为 0，除非该值为 0。
	`void tciSetIntNumberOfDigits (TciValue inst, unsigned long int dig_num)`	在整数值中设置数字的号码。
	`void tciSetIntSign (TciValue inst, Boolean sign)`	把符合设置为+（true）或 -（false）。
	`void tciSetIntDigit (TciValue inst, unsigned long int position, char digit)`	在位置'position'处设置数字的值，position 0 是最低有效位。
FloatValue		
TFloat getFloat()	`double tciGetFloatValue(TciValue inst)`	
void setFloat (in TFloat value)	`void tciSetFloatValue(TciValue inst, double value)`	
BooleanValue		
TBoolean getBoolean()	`Boolean tciGetBooleanValue(TciValue inst)`	
void setBoolean (in TBoolean value)	`void tciSetBooleanValue (TciValue inst, Boolean value)`	

TCI IDL接口	ANSI-C表示法	注释和说明
ObjidValue		
TObjid getObjid()	`TciObjidValue tciGetTciObjidValue(TciValue inst)`	
void setObjid(in TObjid value)	`void tciSetObjidValue(TciValue inst, TciObjidValue value)`	
CharstringValue		
TString getString()	`TciCharStringValue tciGetCStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetCStringValue (TciValue inst, TciCharString value)`	
TChar getChar (in TInteger position)	`char tciGetCStringCharValue (TciValue inst, long int position)`	
void setChar (in TInteger position, in char value)	`void tciSetCStringCharValue (TciValue inst, long int position, char value)`	
TInteger getLength()	`unsigned long int tciGetCStringLength (TciValue inst)`	
void setLength(in TInteger len)	`void tciSetCStringLength (TciValue inst, unsigned long int len)`	
UniversalCharstringValue		
TString getString()	`TciUCStringValuetciGetUCStringValue(TciValue inst)`	
void setString (in TString value)	`void tciSetUCStringValue (TciValue inst, TciUCStringValue value)`	
TUniversalChar getChar (in TInteger position)	`void tciGetUCStringCharValue (TciValue inst, unsigned long int position, TciUCValue result)`	
void setChar (in TInteger position, in TUniversalChar value)	`void tciSetUCStringCharValue (TciValue inst, unsigned long int position, TciUCValue value)`	
TInteger getLength()	`unsigned long int tciGetUCStringLength(TciValue inst)`	
void setLength (in TInteger len)	`void tciSetUCStringLength (TciValue inst, unsigned long int len)`	
BitstringValue		
TString getString()	`String tciGetBStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetBStringValue (TciValue inst, String value)`	
TChar getBit (in integer position)	`int tciGetBStringBitValue (TciValue inst, long int position)`	
void setBit (in TInteger position, in TInteger value)	`void tciSetBStringBitValue (TciValue inst, unsigned long int position, int value)`	
TInteger getLength()	`unsigned long int tciGetBStringLength(TciValue inst)`	
void setLength (in TInteger len)	`void tciSetBStringLength (TciValue inst, long int len)`	
HexstringValue		
TString getString()	`String tciGetHStringValue(TciValue inst)`	
void setString (in TString value)	`void tciSetHStringValue (TciValue inst, String value)`	
TChar getHex (in TInteger position)	`int tciGetHStringHexValue (TciValue inst, unsigned long int position)`	

TCI IDL接口	ANSI-C表示法	注释和说明

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

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

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

OctetstringValue		
TString getString()	`String tciGetOStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetOStringValue` `(TciValue inst, String value)`	
TChar getOctet(in TInteger position)	`int tciGetOStringOctetValue` `(TciValue inst, unsigned long int position)`	

void setOctet
(in TInteger position,
in TInteger value)
	`void tciSetOStringOctetValue`
	`(TciValue inst,`
	`unsigned long int position,`
	`int value)`	

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

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

RecordValue		

Value getField (in TString fieldName)	`TciValue tciGetRecFieldValue`
	`(TciValue inst,`
	`String fieldName)`	

void setField
(in TString fieldName,
in Value value)
	`void tciSetRecFieldValue`
	`(TciValue inst,`
	`String fieldName,`
	`TciValue value)`	

TString[] getFieldNames()	`char** tciGetRecFieldNames(TciValue inst)`	返回字段名称的 NULL 终接阵列。
RecordOfValue		

Value getField (in TInteger position)	`TciValue tciGetRecOfFieldValue`
	`(TciValue inst,`
	`unsigned long int position)`	

void setField
(in TInteger position,
in Value value)
	`void tciSetRecOfFieldValue`
	`(TciValue inst,`
	`unsigned long int position,`
	`TciValue value)`	

void appendField (in Value value)	`void tciAppendRecOfFieldValue`
	`(TciValue inst,`
	`TciValue value)`	

Type getElementType()	`TciType tciGetRecOfElementType(TciValue inst)`	
TInteger getLength()	`unsigned long int tciGetRecOfLength(TciValue inst)`	

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

UnionValue		

Value getVariant
(in TString variantName)
	`TciValue tciGetUnionVariant`
	`(TciValue inst,`
	`String variantName)`	

void setVariant
(in TString variantName,
in Value value)
	`void tciSetUnionVariant`
	`(TciValue inst,`
	`String variantName,`
	`TciValue value)`	

TString getPresentVariantName()	`String tciGetUnionPresentVariantName` `(TciValue inst)`	
TString[] getVariantNames()	`char** tciGetUnionVariantNames(TciValue inst)`	返回字段名称的 NULL 终接阵列。

TCI IDL接口	ANSI-C表示法	注释和说明
EnumeratedValue		
TString getEnum()	`String tciGetEnumValue(TciValue inst)`	
void setEnum (in TString enumValue)	`void tciSetEnumValue (TciValue inst, String enumValue)`	
VerdictValue		
TInteger getVerdict()	`int tciGetVerdictValue(TciValue inst)`	
void setVerdict (in TInteger verdict)	`void tciSetVerdictValue(TciValue inst, int verdict)`	
`AddressValue`		
Value getAddress()	`TciValue tciGetAddressValue(TciValue inst)`	
void setAddress (in Value value)	`void tciSetAddressValue(TciValue inst, TciValue value)`	

9.3 登录接口

TCI IDL接口	ANSI-C表示法	注释和说明
TciValueTemplate		
TBoolean isOmit()	`Boolean tciIsOmit(TciValueTemplate inst)`	从 IDL 再使用的布尔类型（OMG 建议）
TBoolean isAny()	`Boolean tciIsAny(TciValueTemplate inst)`	从 IDL 再使用的布尔类型（OMG 建议）
TBoolean isAnyOrOmit()	`Boolean tciIsAnyOrOmit(TciValueTemplate inst)`	从 IDL 再使用的布尔类型（OMG 建议）
TString getTemplateDef()	`String tciGetTemplateDef(TciValueTemplate inst)`	从 IDL 再使用的串类型（OMG 建议）
TciNonValueTemplate		
TBoolean isAny()	`Boolean tciIsAny(TciNonValueTemplate inst)`	从 IDL 再使用的布尔类型（OMG 建议）
TBoolean isAll()	`Boolean tciIsAll(TciNonValueTemplate inst)`	从 IDL 再使用的布尔类型（OMG 建议）
TString getTemplateDef()	`String tciGetTemplateDef(TciNonValueTemplate inst)`	从 IDL 再使用的串类型（OMG 建议）

9.4 操作接口

要求的TCI-TM		
void tciRootModule (in TciModuleIdType moduleId)	`void tciRootModule(String moduleId)`	
TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName)	`TciModuleParameterListType tciGetModuleParameters (TciModuleIdType moduleName)`	
TciTestCaseIdListType tciGetTestCases()	`TciTestCaseIdListType tciGetTestCases()`	
TciParameterTypeListType tciGetTestCaseParameters (in TciTestCaseIdType testCaseId)	`TciParameterTypeListType tciGetTestCaseParameters (TciTestCaseIdType testCaseId)`	
TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)	`TriPortIdList tciGetTestCaseTSI (TciTestCaseIdType testCaseId)`	

void tciStartTestCase
(in TciTestCaseIdType testCaseId,
in TciParameterListType parameterlist)
	`void tciStartTestCase`
	`(TciTestCaseIdType testCaseId,`
	`TciParameterListType parameterlist)`	

void tciStopTestCase()	`void tciStopTestCase()`	
TriComponentId tciStartControl()	`TriComponentId tciStartControl()`	
void tciStopControl()	`void tciStopControl()`	
提供的TCI-TM		

void tciTestCaseStarted
(in TciTestCaseIdType testCaseId, in
TciParameterListType parameterList, in Tfloat
timer)
	`void tciTestCaseStarted`
	`(TciTestCaseIdType testCaseId,`
	`TciParameterListType parameterList,`
	`double timer)`	

void tciTestCaseTerminated
(in VerdictValue verdict,
in TciParameterListType parameterlist)
	`void tciTestCaseTerminated`
	`(TciVerdictValue verdict,`
	`TciParameterListType parameterlist)`	

void tciControlTerminated()	`void tciControlTerminated()`	
Value tciGetModulePar (in TciModuleParameterIdType parameterId)	`tciValue tciGetModulePar` `(TciModuleParameterIdType parameterId)`	
void tciLog(in TString message)	`void tciLog(String message)`	
void tciError(in TString message)	`void tciError(String message)`	
要求的TCI-CD		
Type getTypeForName (in String typeName)	`TciType tciGetTypeForName` `(String typeName)`	
Type getInteger()	`TciType tciGetIntegerType()`	
Type getFloat()	`TciType tciGetFloatType()`	
Type getBoolean()	`TciType tciGetBooleanType()`	
Type getChar()	`TciType tciGetCharType()`	
Type getUniversalChar()	`TciType tciGetUniversalCharType()`	
Type getObjid()	`TciType tciGetTciObjidType()`	
Type getCharstring()	`TciType tciGetTciCharstringType()`	
Type getUniversalCharstring()	`TciType` `tciGetUniversalCharstringType()`	
Type getHexstring()	`TciType tciGetHexstringType()`	
Type getBitstring()	`TciType tciGetBitstringType()`	
Type getOctetstring()	`TciType tciGetOctetstringType()`	
Type getVerdict()	`TciType tciGetVerdictType()`	
void tciErrorReq(in String message)	`void tciErrorReq(String message)`	
提供的TCI-CD		

Value decode
(in TriMessageType message,
in Type decodingHypothesis)
	`TciValue tciDecode`
	`(BinaryString message,`
	`TciType decHypothesis)`
		从 TRI 再使用的 BinaryString 类型

TriMessageType encode (in Value value)	`BinaryString tciEncode(TciValue value)`	
要求的TCI-CH		

void tciEnqueueMsgConnected
(in TriPortIdType sender,
in TriComponentIdType receiver,
in Value rcvdMessage)
	`void tciEnqueueMsgConnected`
	`(TriPortId sender,`
	`TriComponentId receiver,`
	`TciValue rcvdMessage)`	

void tciEnqueueCallConnected
(in TriPortIdType sender,
in TriComponentIdType receiver,
in TriSignatureIdType signature,
in TciParameterListType parameterList)
	`void tciEnqueueCallConnected`
	`(TriPortId sender,`
	`TriComponentId receiver,`
	`TriSignatureId signature,`
	`TciParameterListType parameterList)`	

void tciEnqueueReplyConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in TciParameterListType parameterList, in Value returnValue)	void tciEnqueueReplyConnected (TriPortId sender, TriComponentId receiver, TriSignatureId signature, TciParameterListType parameterList, TciValue returnValue)	
void tciEnqueueRaiseConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in Value exception)	void tciEnqueueRaiseConnected (TriPortId sender, TriComponentId receiver, TriSignatureIdType signature, TciValue exception)	
TriComponentIdType tciCreateTestComponent (in TciTestComponentKindType kind in Type componentType, in TString name)	TriComponentId tciCreateTestComponent (TciTestComponentKindType kind, TciType componentType, String name)	
void tciStartTestComponent (in TriComponentIdType component, in TciBehaviourIdType behavior, in TciParamaterListType parameterList)	void tciStartTestComponent (TriComponentId component, TciBehaviourIdType behavior, TciParamaterListType parameterList)	
void tciStopTestComponent (in TriComponentIdType component)	void tciStopTestComponent (TriComponentId component)	
void tciConnect (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciConnect (TriPortId fromPort, TriPortId toPort)	
void tciDisconnect (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciDisconnect (TriPortId fromPort, TriPortId toPort)	
void tciMap (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciMap (TriPortId fromPort, TriPortId toPort)	
void tciUnmap (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciUnmap (TriPortId fromPort, TriPortId toPort)	
void tciTestComponentTerminated (in TriComponentIdType component, in VerdictValue verdict)	void tciTestComponentTerminated (TriComponentId component, TciVerdictValue verdict)	
boolean tciTestComponentRunning (in TriComponentIdType component)	Boolean tciTestComponentRunning (TriComponentId component)	
boolean tciTestComponentDone (in TriComponentIdType component)	boolean tciTestComponentDone (TriComponentId component)	
TriComponentIdType tciGetMTC()	TriComponentId tciGetMTC()	
void tciExecuteTestCase (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)	void tciExecuteTestCase (TciTestCaseIdType testCaseId, TriPortIdList tsiPortList)	
void tciReset()	void tciReset()	
void tciKillTestComponent (in TriComponentIdType component)	void tciKillTestComponent (TriComponentId component)	
TBoolean tciTestComponentAlive (in TriComponentIdType component)	Boolean tciTestComponentAlive (TriComponentId component)	
TBoolean tciTestComponentKilled (in TriComponentIdType component)	Boolean tciTestComponentKilled (TriComponentId component)	
提供的TCI-CH		
void tciSendConnected (in TriPortIdType sender, in TriComponentIdType receiver, in Value sendMessage)	void tciSendConnected (TriPortId sender, TriComponentId receiver, TciValue sendMessage)	
void tciSendConnectedBC (in TriPortIdType sender, in Value sendMessage)	void tciSendConnectedBC (TriPortId sender, TciValue sendMessage)	

void tciSendConnectedMC (in TriPortIdType sender, in TriComponentIdListType receivers, in Value sendMessage)	void tciSendConnectedMC (TriPortId sender, TriComponentIdList receivers, TciValue sendMessage)	
void tciCallConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in TciParameterListType parameterList)	void tciCallConnected (TriPortId sender, TriComponentId receiver, TriSignatureId signature, TciParameterListType parameterList)	
void tciCallConnectedBC (in TriPortIdType sender, in TriSignatureIdType signature, in TciParameterListType parameterList)	void tciCallConnectedBC (TriPortId sender, TriSignatureId signature, TciParameterListType parameterList)	
void tciCallConnectedMC (in TriPortIdType sender, in TriComponentIdListType receivers, in TriSignatureIdType signature, in TciParameterListType parameterList)	void tciCallConnectedMC (TriPortId sender, TriComponentIdList receivers, TriSignatureId signature, TciParameterListType parameterList)	
void tciReplyConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in TciParameterListType parameterList, in Value returnValue)	void tciReplyConnected (TriPortId sender, TriComponentId receiver, TriSignatureIdType signature, TciParameterListType parameterList, TciValue returnValue)	
void tciReplyConnectedBC (in TriPortIdType sender, in TriSignatureIdType signature, in TciParameterListType parameterList, in Value returnValue)	void tciReplyConnectedBC (TriPortId sender, TriSignatureIdType signature, TciParameterListType parameterList, TciValue returnValue)	
void tciReplyConnectedMC (in TriPortIdType sender, in TriComponentIdListType receivers, in TriSignatureIdType signature, in TciParameterListType parameterList, in Value returnValue)	void tciReplyConnectedMC (TriPortId sender, TriComponentIdList receivers, TriSignatureIdType signature, TciParameterListType parameterList, TciValue returnValue)	
void tciRaiseConnected (in TriPortIdType sender, in TriComponentIdType receiver, in TriSignatureIdType signature, in Value exception)	void tciRaiseConnected (TriPortId sender, TriComponentId receiver, TriSignatureId signature, TciValue exception)	
void tciRaiseConnectedBC (in TriPortIdType sender, in TriSignatureIdType signature, in Value exception)	void tciRaiseConnectedBC (TriPortId sender, TriSignatureId signature, TciValue exception)	
void tciRaiseConnectedMC (in TriPortIdType sender, in TriComponentIdListType receivers, in TriSignatureIdType signature, in Value exception)	void tciRaiseConnectedMC (TriPortId sender, TriComponentIdList receivers, TriSignatureId signature, TciValue exception)	
TriComponentIdType tciCreateTestComponentReq (in TciTestComponentKindType kind, in Type componentType, in TString name)	TriComponentId tciCreateTestComponentReq (TciTestComponentKindType kind, TciType componentType, String name)	
void tciStartTestComponentReq (in TriComponentIdType component, in TciBehaviourIdType behavior, in TciParamaterListType parameterList)	void tciStartTestComponentReq (TriComponentId component, TciBehaviourIdType behavior, TciParamaterListType parameterList)	
void tciStopTestComponentReq (in TriComponentIdType component)	void tciStopTestComponentReq (TriComponentId component)	
void tciConnectReq (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciConnectReq (TriPortId fromPort, TriPortId toPort)	

void tciDisconnectReq (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciDisconnectReq (TriPortId fromPort, TriPortId toPort)	
void tciMapReq (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciMapReq (TriPortId fromPort, TriPortId toPort)	
void tciUnmapReq (in TriPortIdType fromPort, in TriPortIdType toPort)	void tciUnmapReq (TriPortId fromPort, TriPortId toPort)	
void tciTestComponentTerminatedReq (in TriComponentIdType component, in VerdictValue verdict)	void tciTestComponentTerminatedReq (TriComponentId component, TciVerdictValue verdict)	
boolean tciTestComponentRunningReq (in TriComponentIdType component)	Boolean tciTestComponentRunningReq (TriComponentId component)	
boolean tciTestComponentDoneReq (in TriComponentIdType component)	Boolean tciTestComponentDoneReq (TriComponentId component)	
TriComponentIdType tciGetMTCReq()	TriComponentId tciGetMTCReq()	
void tciExecuteTestCaseReq (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)	void tciExecuteTestCaseReq (TciTestCaseIdType testCaseId, TriPortIdList tsiPortList)	
void tciResetReq()	void tciResetReq()	
void tciKillTestComponentReq (in TriComponentIdType component)	void tciKillTestComponentReq (TriComponentId component)	
TBoolean tciTestComponentAliveReq (in TriComponentIdType component)	TBoolean tciTestComponentAliveReq (TriComponentId component)	
TBoolean tciTestComponentKilledReq (in TriComponentIdType component)	TBoolean tciTestComponentKilledReq (TriComponentId component)	
提供的TCI-TL		
void tliTcExecute (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	void tliTcExecute (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId, TriParameterList pars, TriTimerDuration dur)	
void tliTcStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	void tliTcStart (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId, TriParameterList pars, TriTimerDuration dur)	
void tliTcStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliTcStop (String am, int ts, String src, int line, TriComponentId c)	
void tliTcStarted (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	void tliTcStarted (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId, TriParameterList pars, TriTimerDuration dur)	
void tliTcTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in VerdictValue outcome)	void tliTcTerminated (String am, int ts, String src, int line, TriComponentId c, TciTestCaseIdType tcId, TriParameterList pars, VerdictValue outcome)	
void tliCtrlStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliCtrlStart (String am, int ts, String src, int line, TriComponentId c)	

void tliCtrlStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliCtrlStop (String am, int ts, String src, int line, TriComponentId c)	
void tliCtrlTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliCtrlTerminated (String am, int ts, String src, int line, TriComponentId c)	
void tliMSend_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressType address, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	void tliMSend_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriAddress address, TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure)	
void tliMSend_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	void tliMSend_m_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure)	
void tliMSend_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	void tliMSend_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriAddressList addresses, TriStatus encoderFailure, TriMessage msg, TriStatus transmissionFailure)	
void tliMSend_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	void tliMSend_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriComponentId to, TriStatus transmissionFailure)	
void tliMSend_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType transmissionFailure)	void tliMSend_c_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriStatus transmissionFailure)	
void tliMSend_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	void tliMSend_c_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriComponentIdList toList, TriStatus transmissionFailure)	
void tliMDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriMessageType msg, in TriAddressType address)	void tliMDetected_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriMessage msg, TriAddress address)	
void tliMDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType from)	void tliMDetected_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TriComponentId from)	

void tliMMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliMMismatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs, TriAddress address, TciValueTemplate addressTmpl)	
void tliMMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliMMismatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TciValueTemplate msgTmpl, TciValueDifferenceList diffs, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliMReceive_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliMReceive_m(String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TciValueTemplate msgTmpl, TriAddress address, TciValueTemplate addressTmpl)	
void tliMReceive_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliMReceive_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, Value msgValue, TciValueTemplate msgTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrCall_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	void tliPrCall_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriAddress address, TriStatus encoderFailure, TriParameterList pars, TriStatus transmissionFailure)	
void tliPrCall_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	void tliPrCall_m_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriStatus encoderFailure, TriParameterList pars, TriStatus transmissionFailure)	
void tliPrCall_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	void tliPrCall_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriAddressList addresses, TriStatus encoderFailure, TriParameterList pars, TriStatus transmissionFailure)	

void tliPrCall_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	void tliPrCall_c (String am, int ts, String src int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriComponentId to, TriStatus transmissionFailure)	
void tliPrCall_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType transmissionFailure)	void tliPrCall_c_BC (String am, int ts, String src int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriStatus transmissionFailure)	
void tliPrCall_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	void tliPrCall_c_MC (String am, int ts, String src int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TriComponentIdList toList, TriStatus transmissionFailure)	
void tliPrGetCallDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterListType pars, in TriAddressType address)	void tliPrGetCallDetected_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TriParameterList pars, TriAddress address)	
void tliPrGetCallDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType from)	void tliPrGetCallDetected_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterList parsValue, TriComponentId from)	
void tliPrGetCallMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrGetCallMismatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TciValueTemplate parsTmpl, TciValueDifferenceList diffs, TriAddress address, TciValueTemplate addressTmpl)	
void tliPrGetCallMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrGetCallMismatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TciValueTemplate parsTmpl, TciValueDifferenceList diffs, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrGetCall_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrGetCall_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TciValueTemplate parsTmpl, TriAddress address, TciValueTemplate addressTmpl)	

void tliPrGetCall_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrGetCall_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, TciValueTemplate parsTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrReply_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	void tliPrReply_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureIdType signature, TciParameterList parsValue, Value replValue, TriAddress address, TriStatus encoderFailure, TriParameter repl, TriStatus transmissionFailure)	
void tliPrReply_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	void tliPrReply_m_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureIdType signature, TciParameterList parsValue, Value replValue, TriStatus encoderFailure, TriParameter repl, TriStatus transmissionFailure)	
void tliPrReply_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	void tliPrReply_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureIdType signature, TciParameterList parsValue, Value replValue, TriAddressList addresses, TriStatus encoderFailure, TriParameter repl, TriStatus transmissionFailure)	
void tliPrReply_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	void tliPrReply_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TriComponentId to, TriStatus transmissionFailure)	
void tliPrReply_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriStatusType transmissionFailure)	void tliPrReply_c_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TriStatus transmissionFailure)	
void tliPrReply_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	void tliPrReply_c_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TriComponentIdList toList, TriStatus transmissionFailure)	

void tliPrGetReplyDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterType repl, in TriAddressType address)	void tliPrGetReplyDetected_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TriParameter repl, TriAddress address)	
void tliPrGetReplyDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value replValue, in TriComponentIdType from)	void tliPrGetReplyDetected_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, Value replValue, TriComponentId from)	
void tliPrGetReplyMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrGetReplyMismatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TciValueTemplate replyTmpl, TciValueDifferenceList diffs, TriAddress address, TciValueTemplate addressTmpl)	
void tliPrGetReplyMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrGetReplyMismatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TciValueTemplate replyTmpl, TciValueDifferenceList diffs, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrGetReply_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrGetReply_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TciValueTemplate replyTmpl, TriAddress address, TciValueTemplate addressTmpl)	
void tliPrGetReply_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrGetReply_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value replValue, TciValueTemplate replyTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrRaise_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressType address, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	void tliPrRaise_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriAddress address, TriStatus encoderFailure, TriException exc, TriStatus transmissionFailure)	

void tliPrRaise_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	void tliPrRaise_m_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriStatus encoderFailure, TriException exc, TriStatus transmissionFailure)	
void tliPrRaise_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	void tliPrRaise_m_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriAddressList addresses, TriStatus encoderFailure, TriException exc, TriStatus transmissionFailure)	
void tliPrRaise_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	void tliPrRaise_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriComponentId to, TriStatus transmissionFailure)	
void tliPrRaise_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType transmissionFailure)	void tliPrRaise_c_BC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriStatus transmissionFailure)	
void tliPrRaise_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	void tliPrRaise_c_MC (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TriComponentIdList toList, TriStatus transmissionFailure)	
void tliPrCatchDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)	void tliPrCatchDetected_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TriException exc, TriAddress address)	
void tliPrCatchDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value excValue, in TriComponentIdType from)	void tliPrCatchDetected_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, Value excValue, TriComponentId from)	

void tliPrCatchMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrCatchMismatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, TriAddress address, TciValueTemplate addressTmpl)	
void tliPrCatchMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrCatchMismatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TciValueTemplate excTmpl, TciValueDifferenceList diffs, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrCatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	void tliPrCatch_m (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TciValueTemplate excTmpl, TriAddress address, TciValueTemplate addressTmpl)	
void tliPrCatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	void tliPrCatch_c (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue, Value excValue, TciValueTemplate excTmpl, TriComponentId from, TciNonValueTemplate fromTmpl)	
void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)	void tliPrCatchTimeoutDetected (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature)	
void tliPrCatchTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue)	void tliPrCatchTimeout (String am, int ts, String src, int line, TriComponentId c, TriPortId port, TriSignatureId signature, TciParameterListType parsValue)	
void tliCCreate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TString name)	void tliCCreate (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp, String name)	
void tliCStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciBehaviourIdType name, in TciParameterListType parsValue)	void tliCStart (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp, TciBehaviourIdType name, TciParameterListType parsValue)	

void tliCRunning (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)	void tliCRunning (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp, TBoolean status)	
void tliCStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	void tliCStop (String am, int ts, String src, int line, TriComponentId c, TriComponentId comp)	
void tliCDoneMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	void tliCDoneMismatch (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate compTmpl)	
void tliCDone (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	void tliCDone (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate compTmpl)	
void tliCTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	void tliCTerminated (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)	
void tliPConnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	void tliPConnect (String am, int ts, String src, int line, TriComponentId c, TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2)	
void tliPDisconnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	void tliPDisconnect (String am, int ts, String src, int line, TriComponentId c, TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2)	
void tliPMap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	void tliPMap (String am, int ts, String src, int line, TriComponentId c, TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2)	
void tliPUnmap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	void tliPUnmap (String am, int ts, String src, int line, TriComponentId c, TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2)	
void tliPClear (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	void tliPClear (String am, int ts, String src, int line, TriComponentId c, TriPortId port)	
void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	void tliPStart (String am, int ts, String src, int line, TriComponentId c, TriPortId port)	
void tliPStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	void tliPStop (String am, int ts, String src, int line, TriComponentId c, TriPortId port)	

void tliPHalt (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	void tliPHalt (String am, int ts, String src, int line, TriComponentId c, TriPortId port)	
void tliEncode (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType encoderFailure, in TriMessageType msg, in TString codec)	void tliEncode (String am, int ts, String src, int line, TriComponentId c, Value val, TriStatus encoderFailure, TriMessage msg, String codec)	
void tliDecode (in TString am, in TInteger ts, in TString src, in TInteger l ine, in TriComponentIdType c, in TriMessageType msg, in TriStatusType decoderFailure, in Value val, in TString codec)	void tliDecode (String am, int ts, String src, int l ine, TriComponentId c, TriMessage msg, TriStatus decoderFailure, Value val, String codec)	
void tliTTimeoutDetected (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)	void tliTTimeoutDetected (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer)	
void tliTTimeoutMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)	void tliTTimeoutMismatch (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate timerTmpl)	
void tliTTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)	void tliTTimeout (String am, int ts, String src, int line, TriComponentId c, TciNonValueTemplate timerTmpl)	
void tliTStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)	void tliTStart (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, TriTimerDuration dur)	
void tliTStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)	void tliTStop (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer)	
void tliTRead (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType elapsed)	void tliTRead (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, TriTimerDuration elapsed)	
void tliTRunning (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status)	void tliTRunning (String am, int ts, String src, int line, TriComponentId c, TriTimerId timer, Boolean status)	
void tliSEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType parsValue, in TString kind)	void tliSEnter (String am, int ts, String src, int line, TriComponentId c, String name, TciParameterListType parsValue, String kind)	
void tliSLeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value returnValue, in TString kind)	void tliSLeave (String am, int ts, String src, int line, TriComponentId c, String name, Value returnValue, String kind)	
void tliVar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value varValue)	void tliVar (String am, int ts, String src, int line, TriComponentId c, String name, Value varValue)	

void tliModulePar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value parValue)	void tliModulePar (String am, int ts, String src, int line, TriComponentId c, String name, Value parValue)	
void tliGetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	void tliGetVerdict (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)	
void tliSetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	void tliSetVerdict (String am, int ts, String src, int line, TriComponentId c, VerdictValue verdict)	
void tliLog (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciValueList log)	void tliLog (String am, int ts, String src, int line, TriComponentId c, Value[] log)	
void tliAEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliAEnter (String am, int ts, String src, int line, TriComponentId c)	
void tliALeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliALeave (String am, int ts, String src, int line, TriComponentId c)	
void tliADefaults (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliADefaults (String am, int ts, String src, int line, TriComponentId c)	
void tliAActivate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType pars, in Value ref)	void tliAActivate (String am, int ts, String src, int line, TriComponentId c, String name, TciParameterListType pars, Value ref)	
void tliADeactivate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value ref)	void tliADeactivate (String am, int ts, String src, int line, TriComponentId c, Value ref)	
void tliANomatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliANomatch (String am, int ts, String src, int line, TriComponentId c)	
void tliARepeat (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliARepeat (String am, int ts, String src, int line, TriComponentId c)	
void tliAWait (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	void tliAWait (String am, int ts, String src, int line, TriComponentId c)	

9.5 数据

TCI IDL ADT	ANSI-C表示法（类型定义）	注释和说明
TciModuleIdType	**QualifiedName**	
TciModuleParameterType	typedef struct TciModuleParameterType { String parName; TciValue defaultValue; } TciModuleParameterType;	
TciModuleParameterListType	typedef struct TciModuleParameterListType { long int length; TciModuleParameterType *modParList; } TciModuleParameterListType;	

TCI IDL ADT	ANSI-C表示法（类型定义）	注释和说明
TciModuleIdType	**QualifiedName**	

TciParameterType	typedef struct TciParameterType
	{
	  String parName;
	  TciParameterPassingModeType
	parPassMode;
	  TciValue parValue;
	} TciParameterType;	

TciParameterPassingModeType	typedef enum
	{
	  TCI_IN_PAR = 0,
	  TCI_INOUT_PAR = 1,
	  TCI_OUT_PAR = 2
	} TciParameterPassingModeType;	

TciParameterListType	typedef struct TciParameterListType
	{
	  long int length;
	  TciParameterType *parList;
	} TciParameterListType;
		长度 0 应被解释为“空的列表”。

TciParameterTypeListType	typedef struct TciParameterTypeListType
	{
	  long int length;
	  TciType *parList;
	} TciParameterTypeListType;
		长度 0 应被解释为“空的列表”。

TciTestCaseIdListType	typedef struct TciTestCaseIdListType
	{
	  long int length;
	  QualifiedName *idList;
	} TciTestCaseIdListType;
		长度 0 应被解释为“空的列表”。

TciTypeClassType	typedef enum
	{
	  TCI_ADDRESS_TYPE,
	  TCI_ANYTYPE_TYPE,
	  TCI_BITSTRING_TYPE,
	  TCI_BOOLEAN_TYPE,
	  TCI_CHAR_TYPE,
	  TCI_CHARSTRING_TYPE,
	  TCI_COMPONENT_TYPE,
	  TCI_ENUMERATED_TYPE,
	  TCI_FLOAT_TYPE,
	  TCI_HEXSTRING_TYPE,
	  TCI_INTEGER_TYPE,
	  TCI_OBJID_TYPE,
	  TCI_OCTETSTRING_TYPE,
	  TCI_RECORD_TYPE,
	  TCI_RECORD_OF_TYPE,
	  TCI_SET_TYPE,
	  TCI_SET_OF_TYPE,
	  TCI_UNION_TYPE,
	  TCI_UNIVERSAL_CHAR_TYPE,
	  TCI_UNIVERSAL_CHARSTRING_TYPE,
	  TCI_VERDICT_TYPE
	} TciTypeClassType;	

TciTestComponentKindType	typedef enum
	{
	  TCI_CTRL_COMP,
	  TCI_MTC_COMP,
	  TCI_PTC_COMP,
	  TCI_SYS_COMP
	} TciTestComponentKindType;	

TciBehaviourIdType	QualifiedName	

TciValueDifference	typedef struct TciValueDifference
	{
	  TciValue val;
	  TciValueTemplate tmpl;
	  String desc;
	} TciValueDifference;	

TciValueDifferenceList	typedef struct TciValueDifferenceList
	{
	  long int length;
	  TciValueDifference[] diffList;
	} TciValueDifferenceList;
		长度 0 应被解释为“空的列表”。

9.6 其他

TCI概念	ANSI-C表示法	注释和说明
Verdict 表示法		
NONE	`const int TCI_VERDICT_NONE = 0`	由于以整数规定 TciVerdictValue 接口，因此协商一致必须建立在哪些值规定，哪些值判定的基础上。
PASS	`const int TCI_VERDICT_PASS = 1`	
INCONC	`const int TCI_VERDICT_INCONC = 2`	
FAIL	`const int TCI_VERDICT_FAIL = 3`	
ERROR	`const int TCI_VERDICT_ERROR = 4`	
Objid 表示法		
Objid	`typedef struct TciObjidValue { long int length; TciObjidElem *elements; } TciObjidValue;`	由于 Objid 值通过 Objid 值接口被返回"as is"，因此必须规定一个表示法。
TciObjidElem	`typedef struct TciObjidElemValue { char* elem_as_ascii; long int elem_as_number; void* aux; } TciObjidElemValue;`	
CharstringValue 表示法		
TciCharString	`typedef struct TciCharStringValue { unsigned long int length; char* string; } TciCharStringValue`	
`Universal Character[string] representation`		
Universal Char	`typedef unsigned char[4] TciUCValue`	
Universal Charstring	`typedef struct TciUCStringValue { unsigned long int length; TciUCType *string; } TciUCStringValue;`	

10 W3C XML映射

10.1 引言

本节介绍了用于 TCI 登录接口的 TCI XML 映射[5]、[6]和[7]。用于登录接口的 XML 映射规定了在第 7 节描述的 IDL 定义如何被映射到 XML 上。用于该映射方案的定义在附件 B 中给出。

10.2 范围

IDL 模块 **`tciInterface`** 映射到一个 XML 方案中，名称为 `http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI`。

该方案使用进一步的计划为：

- http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes 对于简单类型映射到XML。
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Types 对于结构型类型映射到XML。
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Values 对于值映射到XML。
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates 对于模板映射到XML。
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Events 对于记录事件映射到XML。

10.3 类型映射

10.3.1 简单类型的映射

10.3.1.1 TBoolean

IDL **TBoolean** 类型映射到 xsd 基本类型 boolean。

10.3.1.2 TString

IDL **TString** 类型映射到 xsd 基本类型 string。

10.3.1.3 TInteger

IDL **TInteger** 类型映射到 xsd 基本类型 integer。

10.3.1.4 TriTimerDurationType

IDL **TriTimerDurationType** 类型映射到 xsd 基本类型 float。

10.3.1.5 TciParameterPassingModeType

IDL **TciParameterPassingModeType** 类型映射到 xsd 基本类型 string，枚举值为'in'、'out'和'inout'。

10.3.1.6 TriStatusType

IDL **TriStatusType** 类型映射到 xsd 基本类型 string，枚举值为'TRI_Ok'和 'TRI_Error'。

10.3.1.7 TciStatusType

IDL **TriStatusType** 类型映射到 xsd 基本类型 string，枚举值为'TCI_Ok'和'TCI_Error'。

10.3.2 复合类型映射

10.3.2.1 TriPortIdType

TriPortIdType 映射到下列复合类型：

```
<xsd:complexType name="TriPortIdType">
    <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="1"  maxOccurs="1"/>
        <xsd:element name="port" type="Types:Port"  minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- comp　TRI部件标识符；
- port　端口的识别。

属性：

- 无。

10.3.2.2 TriComponentIdType

TriComponentIdType 映射到下列复合类型：

```
<xsd:complexType name="TriComponentIdType">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="null"/>
            <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- id　TRI部件标识符；
- null　null标识符。没有TRI部件标识符时使用。

属性：

- 无。

10.3.2.3 TriComponentIdListType

TriComponentIdListType 映射到下列复合类型：

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

单元：

- comp　在该列表中TRI部件的标识符。

属性：

- 无。

10.3.2.4 端口

Port 映射到下列复合类型：

```
<xsd:complexType name="Port">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="index" type="xsd:int" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- id　端口标识符；
- port　端口索引。

属性：

- 无。

10.3.2.5 Id

Id 用做识别部件、端口和定时器并映射到下列复合类型：

```
<xsd:complexType name="Id">
    <xsd:sequence>
        <xsd:element name="name" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="id" type="SimpleTypes:TInteger" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="type" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- name　部件、端口或定时器的名称；
- id　部件、端口或定时器的内部表示；
- type　部件、端口或定时器的类型。

属性：

- 无。

10.3.2.6 TriMessageType

TriMessageType 映射到下列复合类型：

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

单元：

- val 已编码的消息。

属性：

- 无。

10.3.2.7 TriParameterType

TriParameterType 映射到下列复合类型：

```
<xsd:complexType name="TriParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="Values:Value" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="SimpleTypes:TString"/>
    <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

单元：

- val 已编码参数。

属性：

- name 参数名称；
- mode 参数通过模式。

10.3.2.8 TriParameterListType

TriParameterListType 映射到下列复合类型：

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

单元：

- par 该列表中的参数。

属性：

- 无。

10.3.2.9 TriAddressType

TriAddressType 映射到下列复合类型：

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

单元：

- val 地址值。

属性：

- 无。

10.3.2.10 TriAddressListType

TriAddressListType 映射到下列复合类型：

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

单元：

- addr 该列表中的地址。

属性：

- 无。

10.3.2.11 TriExceptionType

TriExceptionType 映射到下列复合类型：

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

单元：

- val 例外情况。

属性：

- 无。

10.3.2.12 TriSignatureIdType

TriSignatureIdType 映射到下列复合类型：

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

单元：

- val 标记。

属性：

- 无。

10.3.2.13 TriAddressType

TriAddressType 映射到下列复合类型：

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

单元：

- val 在SUT内的地址。

属性：

- 无。

10.3.2.14 TriTimerIdType

TriTimerIdType 映射到下列复合类型：

```
<xsd:complexType name="TriTimerIdType">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- `id` 定时器的识别。

属性：

- 无。

10.3.2.15 TriTimerDurationType

`TriTimerDurationType` 映射到下列复合类型：

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

单元：

- `val` 定时器时间间隔。

属性：

- 无。

10.3.2.16 QualifiedName

`QualifiedName` 用于完全限定模块参数、变量等并映射到下列复合类型：

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

单元：

- `moduleName` TTCN-3模块的模块名称。
- `baseName` 完全合格的对象名称。

属性：

- 无。

10.3.2.17 TciBehaviourIdType

`TciBehaviourIdType` 映射到下列复合类型：

```
<xsd:complexType name="TciBehaviourIdType">
    <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- `name` 行为的当限名称。

属性：

- 无。

10.3.2.18 TciTestCaseIdType

`TciTestCaseIdType` 映射到下列复合类型：

```
<xsd:complexType name="TciTestCaseIdType">
    <xsd:sequence>
        <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

单元：

- `name` 测试用例的当限名称。

属性：

- 无。

10.3.2.19 TciParameterType

`TciParameterType` 映射到下列复合类型：

```
<xsd:complexType name="TciParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="Values:Value" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="SimpleTypes:TString"/>
    <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>
```

单元：

- `val` 已编码参数。

属性：

- `name` 参数名称。
- `mode` 参数通过模式。

10.3.2.20 TciParameterListType

`TciParameterListType` 映射到下列复合类型：

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

单元序列：

- `par` 该列表中的参数。

属性：

- 无。

10.3.3 抽象值映射

10.3.3.1 值

`Value` 映射到下列复合类型：

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

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

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

单元的选择：

- integer 一个整数值。
- float 一个浮点值。
- boolean 一个布尔值。
- objid 一个objid值。
- verdicttype 一个verdicttype值。
- bitstring 一个bitstring值。
- hexstring 一个hexstring值。
- octetstring 一个octetstring值。
- charstring 一个charstring值。
- universal_charstring 一个通用charstring值。
- record 一个记录值。
- record_of 一个值的记录。
- set 一组值。
- set_of 值的集合。
- enumerated 一个枚举值。
- union 一个集合值。
- anytype 一个anytype值。
- address 一个地址值。

属性：

- name 值的名称，如果知道的话。
- type 值的类型，如果知道的话。
- module 值的模块，如果知道的话。

10.3.3.2 IntegerValue

IntegerValue 映射到下列复合类型：

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

简单的内容：

- base 整数值作为串。
- extension 与Value相同的属性。

10.3.3.3 FloatValue

FloatValue 映射到下列复合类型：

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

简单的内容：

- base 浮点值作为串。
- extension 与Value相同的属性。

10.3.3.4 BooleanValue

BooleanValue 映射到下列复合类型：

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

简单的内容：

- base 布尔值作为串。
- extension 与Value相同的属性。

10.3.3.5 ObjidValue

ObjidValue 映射到下列复合类型：

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

简单的内容：

- base objid值作为串。
- extension 与Value相同的属性。

10.3.3.6 VerdictValue

VerdictValue 映射到下列复合类型：

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

简单的内容：

- base 判定值作为串。
- extension 与Value相同的属性。

10.3.3.7 BitstringValue

BitstringValue 映射到下列复合类型：

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

简单的内容：

- base bitstring值作为串。
- extension 与Value相同的属性。

10.3.3.8 HexstringValue

HexstringValue 映射到下列复合类型：

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

简单的内容：

- base　　hexstring值作为串。
- extension　　与Value相同的属性。

10.3.3.9 OctetstringValue

OctetstringValue 映射到下列复合类型：

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

简单的内容：

- base　　octetstring值作为串。
- extension　　与Value相同的属性。

10.3.3.10 CharstringValue

CharstringValue 映射到下列复合类型：

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

简单的内容：

- base　　charstring值作为串。
- extension　　与Value相同的属性。

10.3.3.11 UniversalCharstringValue

UniversalCharstringValue 映射到下列复合类型：

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

简单的内容：

- base　　通用charstring值作为串。
- extension　　与Value相同的属性。

10.3.3.12 RecordValue

RecordValue 映射到下列复合类型：

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

单元序列：

- `integer` 一个整数值。
- `float` 一个浮点值。
- `boolean` 一个布尔值。
- `objid` 一个objid值。
- `verdicttype` 一个verdicttype值。
- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `anytype` 一个anytype值。
- `address` 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.13 RecordOfValue

RecordOfValue 映射到下列复合类型:

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

单元序列的选择：

- integer 一个整数值。
- float 一个浮点值。
- boolean 一个布尔值。
- objid 一个objid值。
- verdicttype 一个verdicttype值。
- bitstring 一个bitstring值。
- hexstring 一个hexstring值。
- octetstring 一个octetstring值。
- charstring 一个charstring值。
- universal_charstring 一个通用charstring值。
- record 一个记录值。
- record_of 一个值的记录。
- set 一组值。
- set_of 值的集合。
- enumerated 一个枚举值。
- union 一个集合值。
- anytype 一个anytype值。
- address 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.14 SetValue

SetValue 映射到下列复合类型：

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

单元序列：

- integer 一个整数值。
- float 一个浮点值。
- boolean 一个布尔值。
- objid 一个objid值。
- verdicttype 一个verdicttype值。

- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `anytype` 一个anytype值。
- `address` 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.15 SetOfValue

`SetOfValue` 映射到下列复合类型：

```
<xsd:complexType name="SetOfValue">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
```

```
            <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>
```

单元序列的选择：

- `integer` 一个整数值。
- `float` 一个浮点值。
- `boolean` 一个布尔值。
- `objid` 一个objid值。
- `verdicttype` 一个verdicttype值。
- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `anytype` 一个anytype值。
- `address` 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.16 EnumeratedValue

`EnumeratedValue` 映射到下列复合类型：

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

单元序列：

- `element` 枚举值。

属性：

- 与Value相同的属性。

10.3.3.17 UnionValue

`UnionValue` 映射到下列复合类型：

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

单元的选择：

- `integer` 一个整数值。
- `float` 一个浮点值。
- `boolean` 一个布尔值。
- `objid` 一个objid值。
- `verdicttype` 一个verdicttype值。
- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `anytype` 一个anytype值。
- `address` 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.18 AnytypeValue

AnytypeValue 映射到下列复合类型：

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

单元的选择：

- `integer` 一个整数值。
- `float` 一个浮点值。
- `boolean` 一个布尔值。
- `objid` 一个objid值。
- `verdicttype` 一个verdicttype值。
- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `address` 一个地址值。

属性：

- 与Value相同的属性。

10.3.3.19 AddressValue

AddressValue 映射到下列复合类型：

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

```
            <xsd:element name="charstring" type="Values:OctetstringValue"/>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue"/>
            <xsd:element name="record" type="Values:RecordValue"/>
            <xsd:element name="record_of" type="Values:RecordOfValue"/>
            <xsd:element name="set" type="Values:SetValue"/>
            <xsd:element name="set_of" type="Values:SetOfValue"/>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
            <xsd:element name="union" type="Values:UnionValue"/>
            <xsd:element name="anytype" type="Values:AnytypeValue"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
    </xsd:complexType>
```

单元的选择：

- `integer` 一个整数值。
- `float` 一个浮点值。
- `boolean` 一个布尔值。
- `objid` 一个objid值。
- `verdicttype` 一个verdicttype值。
- `bitstring` 一个bitstring值。
- `hexstring` 一个hexstring值。
- `octetstring` 一个octetstring值。
- `charstring` 一个charstring值。
- `universal_charstring` 一个通用charstring值。
- `record` 一个记录值。
- `record_of` 一个值的记录。
- `set` 一组值。
- `set_of` 值的集合。
- `enumerated` 一个枚举值。
- `union` 一个集合值。
- `anytype` 一个anytype值。

属性：

- 与Value相同的属性。

10.3.4 抽象记录类型映射

规定了额外的类型来简化值和模板之间匹配的记录。

10.3.4.1 TciValueTemplate

`TciValueTemplate` 映射到下列复合类型：

```
<xsd:complexType name="TciValueTemplate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Values:Value">
        <xsd:choice>
            <xsd:element name="integer" type="Templates:IntegerTemplate"/>
            <xsd:element name="float" type="Templates:FloatTemplate"/>
            <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
            <xsd:element name="objid" type="Templates:ObjidTemplate"/>
            <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
            <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
            <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
            <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
            <xsd:element name="universal_charstring"
             type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
```

```
            <xsd:element name="union" type="Templates:UnionTemplate"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
            <xsd:element name="address" type="Templates:AddressTemplate"/>
            <xsd:element name="omit" type="Templates:omit"/>
            <xsd:element name="any" type="Templates:any"/>
            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

单元的选择：

- `integer` 一个整数模板。
- `float` 一个浮点模板。
- `boolean` 一个布尔模板。
- `objid` 一个objid模板。
- `verdicttype` 一个verdicttype模板。
- `bitstring` 一个bitstring模板。
- `hexstring` 一个hexstring模板。
- `octetstring` 一个octetstring模板。
- `charstring` 一个charstring模板。
- `universal_charstring` 一个通用charstring模板。
- `record` 一个记录模板。
- `record_of` 一个模板的记录。
- `set` 一组模板。
- `set_of` 一个模板的集合。
- `enumerated` 一个枚举的模板。
- `union` 一个集合模板。
- `anytype` 一个anytype模板。
- `address` 一个地址模板。
- `omit` 一个omit模板。
- `any` 一个any模板。
- `anyoromit` 一个anyoromit模板。
- `templateDef` 一个复合模板定义。

属性：

- 无。

10.3.4.2 TciNonValueTemplate

`TciNonValueTemplate` 映射到下列复合类型：

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

单元的选择：

- `any` 一个any模板。
- `all` 一个all模板。
- `templateDef` 一个复合模板定义。

属性：

- 无。

10.3.4.3 TciValueList

`TciValueList` 映射到下列复合类型：

```
<xsd:complexType name="TciValueList">
    <xsd:sequence>
        <xsd:element name="val" type="Values:Value" minOccurs="1"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

单元序列：

- `val` 在值列表中的值。

属性：

- 无。

10.3.4.4 TciValueDifference

`TciValueDifference` 映射到下列复合类型：

```
<xsd:complexType name="TciValueDifference">
    <xsd:attribute name="desc" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="val" type="SimpleTypes:xpath" use="required"/>
    <xsd:attribute name="tmpl" type="SimpleTypes:xpath" use="required"/>
</xsd:complexType>
```

单元序列：

- `desc` 失配的原因。
- `val` 参考失配值。
- `tmpl` 参考模板。

属性：

- 无。

10.3.4.5 TciValueDifferenceList

`TciValueDifferenceList` 映射到下列复合类型：

```
<xsd:complexType name="TciValueDifferenceList">
    <xsd:sequence>
        <xsd:element name="diff" type="Templates:TciValueDifference" minOccurs="1"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

单元序列：

- `diff` 在值差别列表中的值/模板差别。

属性：

- 无。

10.4 登录接口上操作的映射

在登录接口上提供的每个操作在 XML 中都有一个相应的复合类型定义。这些复合类型定义为事件的扩展。

10.4.1 Event

Event 映射到下列复合类型：

```
<!–所有事件的通常定义 -->
<xsd:complexType name="Event" mixed="true">
    <xsd:sequence>
        <xsd:element name="am" type="SimpleTypes:TString"/>
    </xsd:sequence>
    <xsd:attribute name="ts" type="xsd:time" use="required"/>
    <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

<!-- 用于测试部件、端口和定时器的一般标识符结构 -->
    <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
    <xsd:attribute name="id" type="SimpleTypes:TInteger" use="required"/>
    <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

单元：

- `am` 在日志中用做表示进一步信息的一个消息。

属性：

- `ts` 事件产生的时间。
- `src` 测试规范的源文件。
- `line` 执行请求的行号。
- `name` 产生该事件的部件名称。
- `id` 产生该事件的部件标识符。
- `type` 产生该事件的部件类型。

10.4.2 操作的映射

操作的映射在下表中给出。

提供的TCI-TL	
void tliTcExecute (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	<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="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTcStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	<xsd:complexType name="tliTcStart"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="tcId" type="Types:TciTestCaseIdType"/> <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTcStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliTcStop"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliTcStarted (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)	<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="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTcTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in VerdictValue outcome)	<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="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="outcome" type="Values:VerdictValue"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCtrlStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliCtrlStart"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliCtrlStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliCtrlStop"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliCtrlTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliCtrlTerminated"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliMSend_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressType address, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <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>

提供的TCI-TL	
void tliMSend_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_m_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMSend_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_m_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMSend_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMSend_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_c_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <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>

提供的TCI-TL	
void tliMSend_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	<xsd:complexType name="tliMSend_c_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriMessageType msg, in TriAddressType address)	<xsd:complexType name="tliMDetected_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Types:TriMessageType"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType from)	<xsd:complexType name="tliMDetected_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Types:TriMessageType"/> <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliMMismatch_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliMMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliMMismatch_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value"/> <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/> <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/> <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMReceive_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliMReceive_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/> <xsd:element name="msgTmpl" type="Templates:TciValueTemplate" minOccurs="0"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliMReceive_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliMReceive_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" 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>
void tliPrCall_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrCall_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_m_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCall_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_m_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCall_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/> <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCall_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_c_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrCall_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrCall_c_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/> <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetCallDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterListType pars, in TriAddressType address)	<xsd:complexType name="tliPrGetcallDetected_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="pars" type="Types:TriParameterListType"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetCallDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType from)	<xsd:complexType name="tliPrGetcallDetected_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="pars" type="Types:TciParameterListType"/> <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetCallMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrGetcallMismatch_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="pars" type="Types:TriParameterListType"/> <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrGetCallMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrGetcallMismatch_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="pars" type="Types:TciParameterListType"/> <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetCall_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrGetcall_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="pars" type="Types:TriParameterListType"/> <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetCall_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrGetcall_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="pars" type="Types:TciParameterListType"/> <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrReply_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrReply_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_m_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrReply_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_m_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <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>

提供的TCI-TL	
void tliPrReply_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrReply_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_c_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrReply_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrReply_c_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetReplyDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterType repl, in TriAddressType address)	<xsd:complexType name="tliPrGetReplyDetected_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="address" type="Types:TriAddressType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrGetReplyDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value replValue, in TriComponentIdType from)	<xsd:complexType name="tliPrGetReplyDetected_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="from" type="Types:TriComponentIdType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetReplyMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrGetReplyMismatch_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="replTmpl" type="Values:Value"/> <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/> <xsd:element name="address" type="Types:TriAddressType"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetReplyMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrGetReplyMismatch_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="replTmpl" type="Values:Value"/> <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/> <xsd:element name="from" type="Types:TriComponentIdType"/> <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrGetReply_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrGetReply_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="replTmpl" type="Values:Value"/> <xsd:element name="address" type="Types:TriAddressType"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrGetReply_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrGetReply_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="replValue" type="Values:Value"/> <xsd:element name="replTmpl" type="Values:Value"/> <xsd:element name="from" type="Types:TriComponentIdType"/> <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrRaise_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressType address, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrRaise_m_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_m_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <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>

提供的TCI-TL	
void tliPrRaise_m_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_m_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/> <xsd:choice> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> <xsd:sequence> <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:choice> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrRaise_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdType to, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrRaise_c_BC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_c_BC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrRaise_c_MC (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)	<xsd:complexType name="tliPrRaise_c_MC"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="excValue" type="Values:Value"/> <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/> <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatchDetected_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)	<xsd:complexType name="tliPrCatchDetected_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="exc" type="Types:TriExceptionType"/> <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatchDetected_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value excValue, in TriComponentIdType from)	<xsd:complexType name="tliPrCatchDetected_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="exc" type="Types:TriExceptionType"/> <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatchMismatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrCatchMismatch_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="exc" type="Values:Value"/> <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/> <xsd:element name="address" type="Types:TriAddressType"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrCatchMismatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrCatchMismatch_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TciParameterListType"/> <xsd:element name="exc" type="Values:Value"/> <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="address" type="Types:TriAddressType"/> <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatch_m (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)	<xsd:complexType name="tliPrCatch_m"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="exception" type="Values:Value"/> <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatch_c (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)	<xsd:complexType name="tliPrCatch_c"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="exception" type="Values:Value"/> <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/> <xsd:element name="from" type="Types:TriComponentIdType"/> <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)	<xsd:complexType name="tliPrCatchTimeoutDetected"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliPrCatchTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue)	<xsd:complexType name="tliPrCatchTimeout"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="port" type="Types:TriPortIdType"/> <xsd:element name="signature" type="Types:TriSignatureIdType"/> <xsd:element name="parsValue" type="Types:TriParameterListType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCCreate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TString name)	<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:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciBehaviourIdType name, in TciParameterListType parsValue)	<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="pars" type="Types:TciParameterListType" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCRunning (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)	<xsd:complexType name="tliCRunning"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="status" type="SimpleTypes:TBoolean"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCAlive, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)	<xsd:complexType name="tliCAlive"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="status" type="SimpleTypes:TBoolean"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	<xsd:complexType name="tliCStop"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliCKill (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)	<xsd:complexType name="tliCKill"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCDoneMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	<xsd:complexType name="tliCDoneMismatch"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> <xsd:attribute name="done" type="SimpleTypes:TBoolean"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCDone (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	<xsd:complexType name="tliCDone"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> <xsd:attribute name="done" type="SimpleTypes:TBoolean"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCKilledMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	<xsd:complexType name="tliCKilledMismatch"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/> </xsd:sequence> <xsd:attribute name="done" type="SimpleTypes:TBoolean"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliCKill (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)	<xsd:complexType name="tliCKill"> <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:attribute name="done" type="SimpleTypes:TBoolean"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliCTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	<xsd:complexType name="tliCTerminated"> without verdict) --> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="comp" type="Types:TriComponentIdType"/> <xsd:element name="verdict" type="Values:VerdictValue" maxOccurs="1"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliPConnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	<xsd:complexType name="tliPConnect"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPDisconnect (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	<xsd:complexType name="tliPDisconnect"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPMap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	<xsd:complexType name="tliPMap"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPUnmap (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)	<xsd:complexType name="tliPUnmap"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPClear (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	<xsd:complexType name="tliPClear"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	<xsd:complexType name="tliPStart"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	<xsd:complexType name="tliPStop"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>
void tliPHalt (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)	<xsd:complexType name="tliPHalt"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:PortConfiguration"/> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliEncode (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType encoderFailure, in TriMessageType msg, in TString codec)	<xsd:complexType name="tliEncode"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="val" type="Values:Value"/> <xsd:choice> <xsd:element name="msg" type="Types:TriMessageType"/> <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/> </xsd:choice> </xsd:sequence> <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliDecode (in TString am, in TInteger ts, in TString src, in TInteger l ine, in TriComponentIdType c, in TriMessageType msg, in TriStatusType decoderFailure, in Value val, in TString codec)	<xsd:complexType name="tliDecode" mixed="true"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:choice> <xsd:element name="val" type="Values:Value"/> <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"/> </xsd:choice> <xsd:element name="msg" type="Types:TriMessageType"/> </xsd:sequence> <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTTimeoutDetected (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)	<xsd:complexType name="tliTTimeoutDetected"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1" minOccurs="1"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTTimeoutMismatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)	<xsd:complexType name="tliTTimeoutMismatch"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1" minOccurs="1"/> <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1" minOccurs="1"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)	<xsd:complexType name="tliTTimeout"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1" minOccurs="1"/> <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1" minOccurs="1"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliTStart (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)	<xsd:complexType name="tliTStart"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType"/> <xsd:element name="dur" type="Types:TriTimerDurationType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTStop (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)	<xsd:complexType name="tliTStop"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTRead (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType elapsed)	<xsd:complexType name="tliTRead"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType"/> <xsd:element name="elapsed" type="Types:TriTimerDurationType"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliTRunning (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status)	<xsd:complexType name="tliTRunning"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="timer" type="Types:TriTimerIdType"/> </xsd:sequence> <xsd:attribute name="status" type="SimpleTypes:TBoolean"/> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliSEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType parsValue, in TString kind)	<xsd:complexType name="tliSEnter"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/> <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="kind" type="SimpleTypes:TString"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliSLeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value returnValue, in TString kind)	<xsd:complexType name="tliSLeave"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/> <xsd:element name="return" type="Values:Value" minOccurs="0"/> <xsd:element name="kind" type="SimpleTypes:TString"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliVar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value varValue)	<xsd:complexType name="tliVar"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/> <xsd:element name="val" type="Values:Value" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliModulePar (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value parValue)	<xsd:complexType name="tliModulePar"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/> <xsd:element name="val" type="Values:Value" minOccurs="0"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliGetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	<xsd:complexType name="tliGetVerdict"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="verdict" type="Values:VerdictValue"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliSetVerdict (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)	<xsd:complexType name="tliSetVerdict"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="verdict" type="Values:VerdictValue"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliLog (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciValueList log)	<xsd:complexType name="tliLog"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="log" type=" Values:Value"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliAEnter (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliAEnter"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliALeave (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliALeave"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliADefaults (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliADefaults"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>

提供的TCI-TL	
void tliAActivate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType pars, in Value ref)	<xsd:complexType name="tliAActivate"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/> <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/> <xsd:element name="ref" type="Values:Value"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliADeactivate (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value ref)	<xsd:complexType name="tliADeactivate"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"> <xsd:sequence> <xsd:element name="ref" type="Values:Value"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>
void tliANomatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliANomatch"> <xsd:complexContent mixed="true"> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliARepeat (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliARepeat"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>
void tliAWait (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)	<xsd:complexType name="tliAWait"> <xsd:complexContent> <xsd:extension base="Events:Event"/> </xsd:complexContent> </xsd:complexType>

11 使用方案

本节包括使用方案，这些方案应该帮助 TCI 的用户和提供 TCI 的用具供货商，来理解在本建议书中规定的操作的语义。

用 UML 序列图来规定方案，序列图显示了 TCI 实体之间的交互。解释了这些方案并且对应于方案有适用的支撑 TTCN-3 段。

11.1 初始化、收集信息、登录

11.1.1 使用方案：初始化

在图 9 中的方案显示了当选择一个 TTCN-3 模块来执行时，一个测试系统的初始化阶段。起初必须设置一个根模块 `tciRootModule`。根模块的模块参数可以获得为 `tciGetModuleParameters`。可以用模块参数信息来询问测试系统用户每个模块参数的具体值。在根模块中可用的测试用例列表可以用 `tciGetTestCases` 重新获得。这些测试用例能从测试管理中直接执行。他们的参数和测试系统接口可以相应地获得为 `tciGetTestCaseParameters` 和 `tciGetTestCaseTSI`。

11.1.1.1 序列图

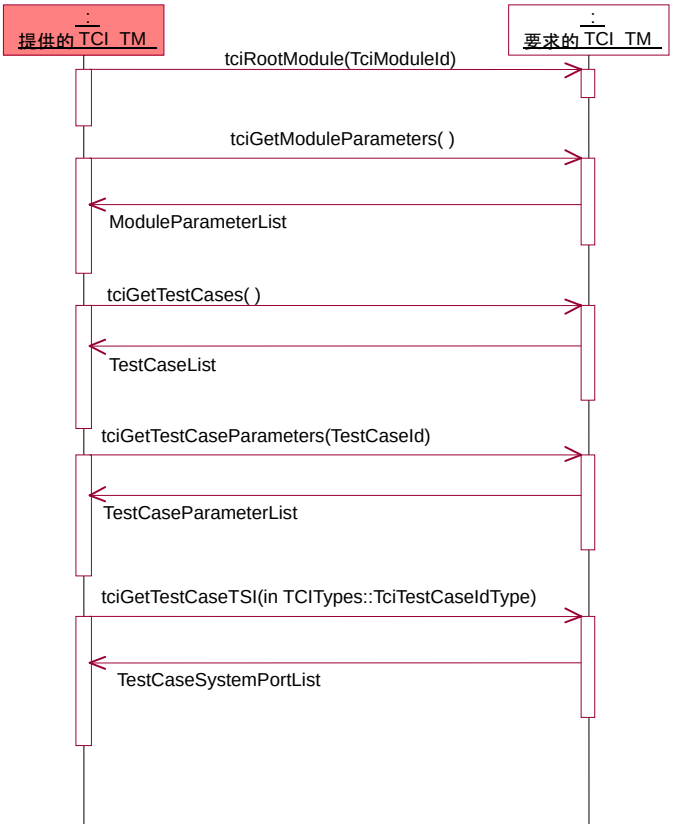

图 9/Z.145－使用方案 — 初始化

11.1.1.2 TTCN-3段

初始化不在 TTCN-3 的范围之内。

11.1.2 使用方案：请求模块参数

在图 10 中的方案显示了一个测试部件如何请求执行测试行为所需要的模块参数的实际值。首先，请求模块参数的类型，然后由 TM 来构建该值并把该值给 TE。

11.1.2.1 序列图

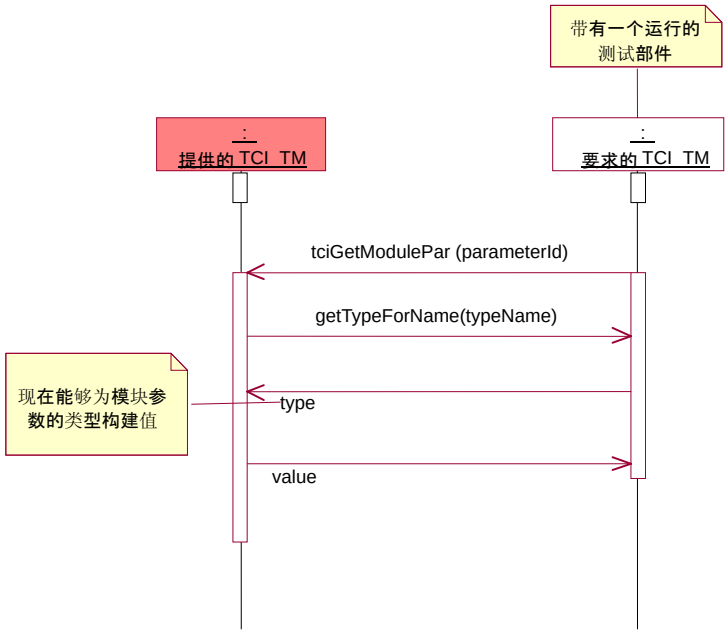

图 10/Z.145－使用方案 — 请求模块参数

11.1.2.2 TTCN-3段

```
module AModule {
    …
    modulepar {
        integer AModulePar
    }
    …
    function AFunction (…) … {
        integer x;
        …
        x:= 2+AModulePar; // an expression with a module parameter
        …
    }
    …
}
```

11.1.3 使用方案：记录

在图 11 中的方案显示了由测试部件执行测试行为期间信息的记录。要记录的消息发送到测试记录上。

11.1.3.1 序列图

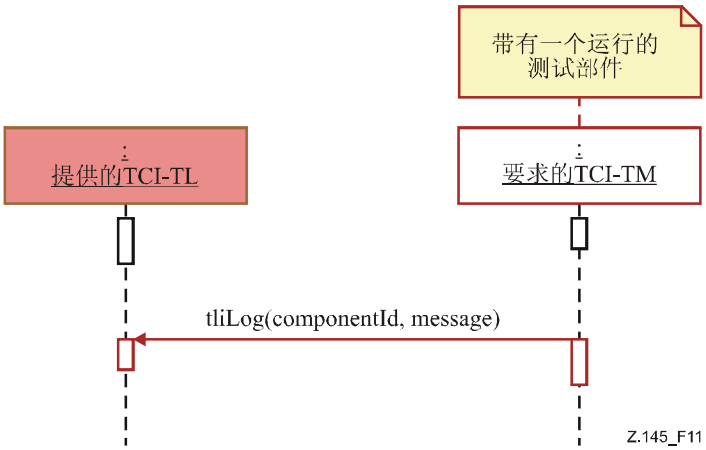

图 11/Z.145－使用方案 — 记录

11.1.3.2 TTCN-3段

```
module AModule {
    …
    function AFunction (…) … {
        …
        log('AMessage');
        …
    }
    …
}
```

11.2 测试用例的执行和控制

11.2.1 使用方案：控制的执行

在图 12 中的方案显示了操作序列要执行 TTCN-3 模块的控制部分。首先选择包括控制部分的模块，然后启动控制，然后执行直到被 TE 终止。

11.2.1.1 序列图

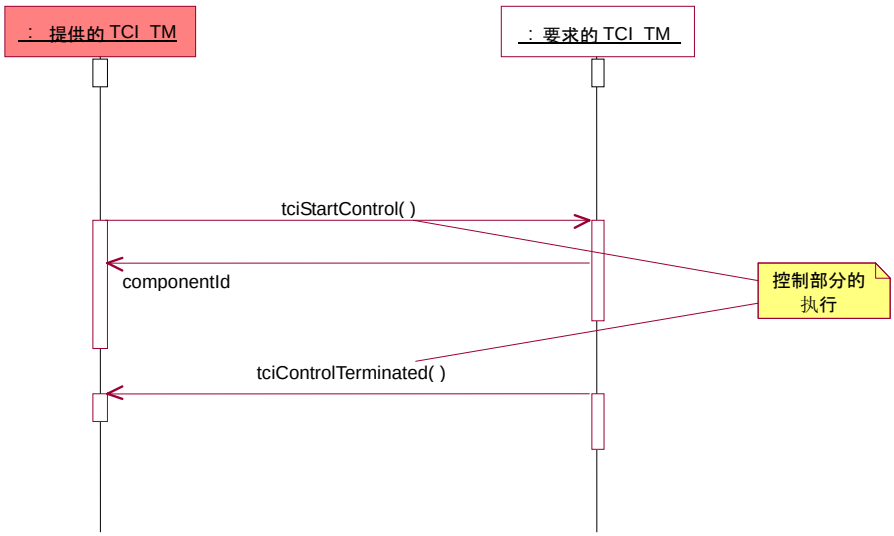

图 12/Z.145－使用方案 — 控制的执行

11.2.1.2 TTCN-3段

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.2.2 使用方案：控制部分内测试用例的执行

在图 13 中的方案显示了在控制部分内如何执行测试用例。

11.2.2.1 序列图

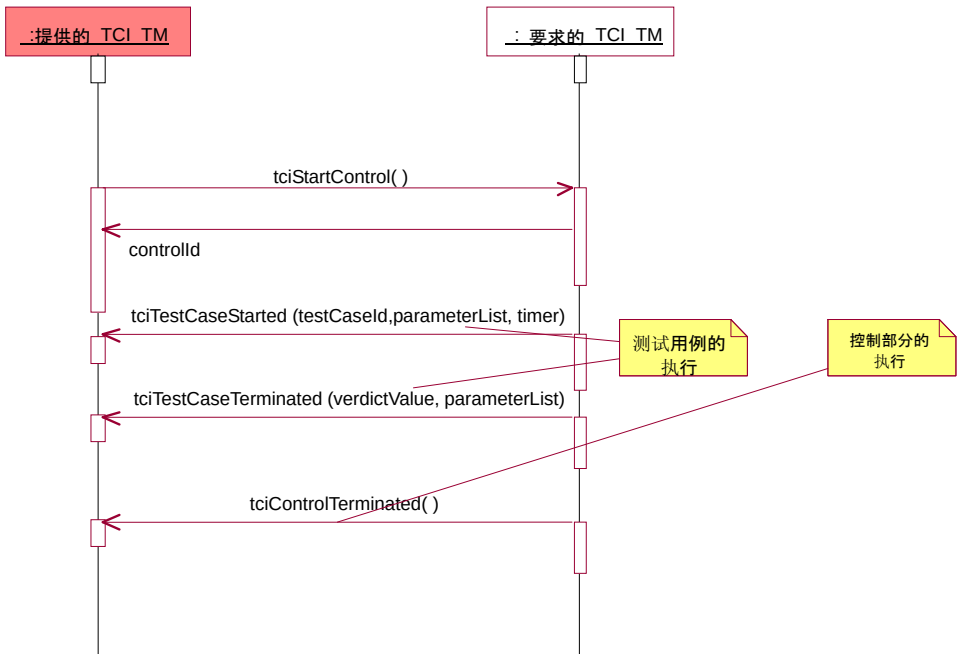

图 13/Z.145－使用方案 — 控制部分内测试用例的执行

11.2.2.2 TTCN-3段

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.2.3 使用方案：直接执行测试用例

在图 14 中的方案显示了如何能从控制部分之外的测试管理直接执行测试用例。在选择了包含要执行的测试用例的 TTCN-3 模块之后，请求启动测试用例。当测试用例完成执行时，TE 通知测试管理终止测试用例。

11.2.3.1 序列图

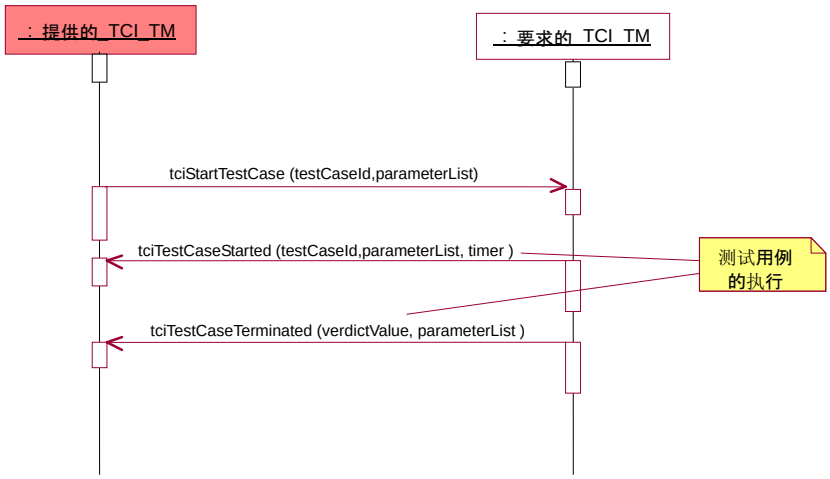

图 14/Z.145－使用方案 — 直接执行测试用例

11.2.3.2 TTCN-3段

直接执行测试用例不在 TTCN-3 的范围之内。

11.2.4 使用方案：执行测试用例到TRI

在图 15 中的方案显示了如何通知 TRI 有关测试用例的执行，以便在需要时能够建立并初始化系统端口。在当前测试用例 MTC 上的测试行为启动之前，必须发出执行测试用例请求。

11.2.4.1 序列图

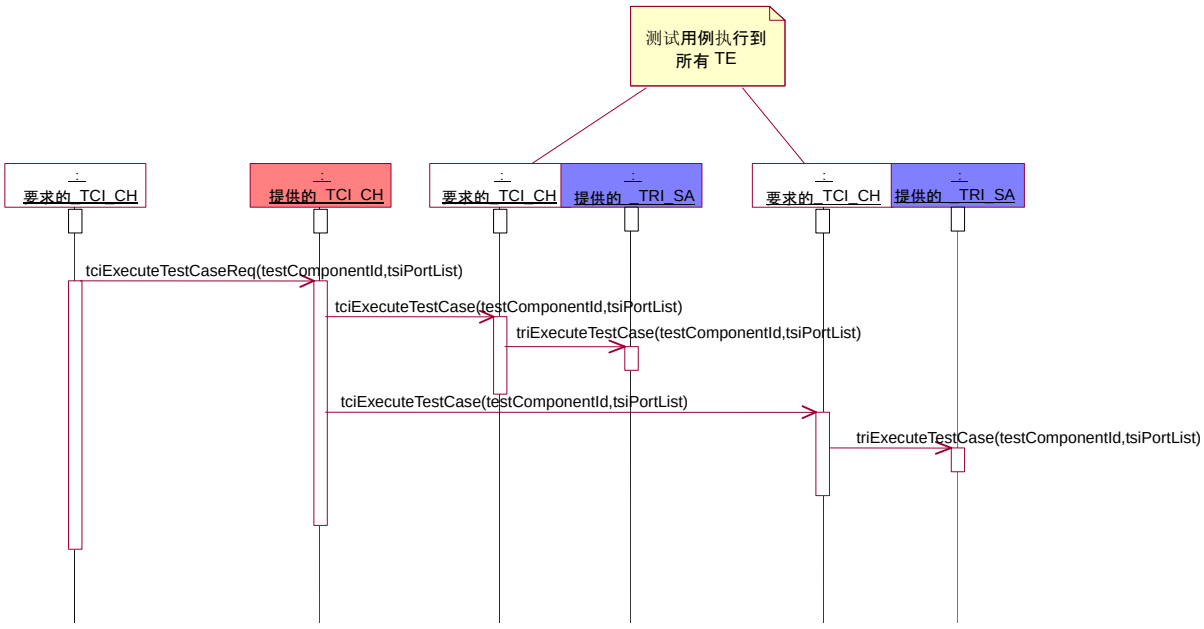

图 15/Z.145－使用方案 — 执行测试用例到TRI

11.2.4.2 TTCN-3段

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3 部件处理

11.3.1 使用方案：创建本地控制部件

在图 16 中的方案显示了在相同节点上创建控制部件，到测试管理 TCI-TM 的用户接口位于该节点上。只要执行 TTCN-3 模块的控制部分，就可以创建控制部件。只要测试管理 TCI-TM 启动控制部分，则把创建测试部件请求发送到 TCI-CH，TCI-CH 再把请求发送到控制部件应被创建的 TE 处。在这种情况下，TE 在相同的节点上。返回控制部件的标识符并发送给 TCI-TM。然后使用标识符来启动控制部件上控制部分的行为。

11.3.1.1 序列图

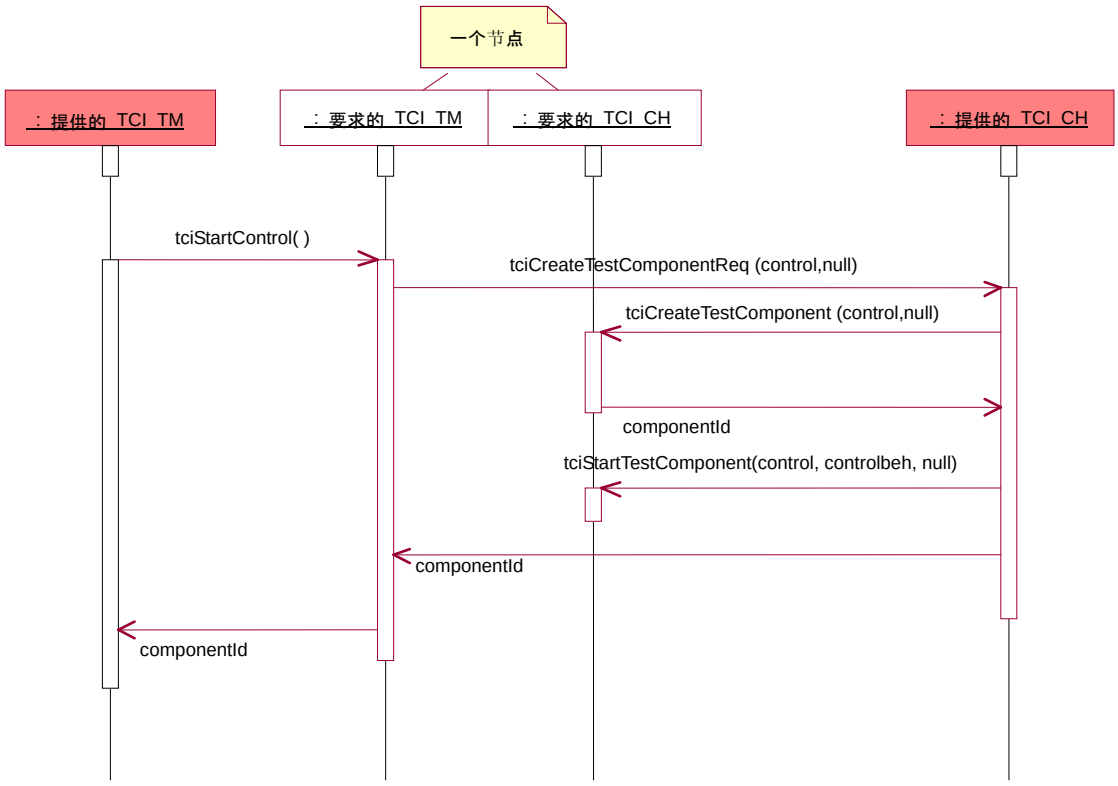

图 16/Z.145－使用方案 — 创建本地控制部件

11.3.1.2 TTCN-3段

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.3.2 使用方案：创建远程控制部件

在图 17 中的方案显示了在另一个节点上创建控制部件，到测试管理 TCI-TM 的用户接口位于该节点上。只要执行 TTCN-3 模块的控制部分，就可以创建控制部件。只要测试管理 TCI-TM 启动控制部分，则把创建测试部件请求发送到 TCI-CH，TCI-CH 再把请求发送到控制部件应被创建的 TE 处。在这种情况下，TE 在另一个远程节点上。返回控制部件的标识符并发送给 TCI-TM。然后使用标识符来启动控制部件上控制部分的行为。

11.3.2.1 序列图

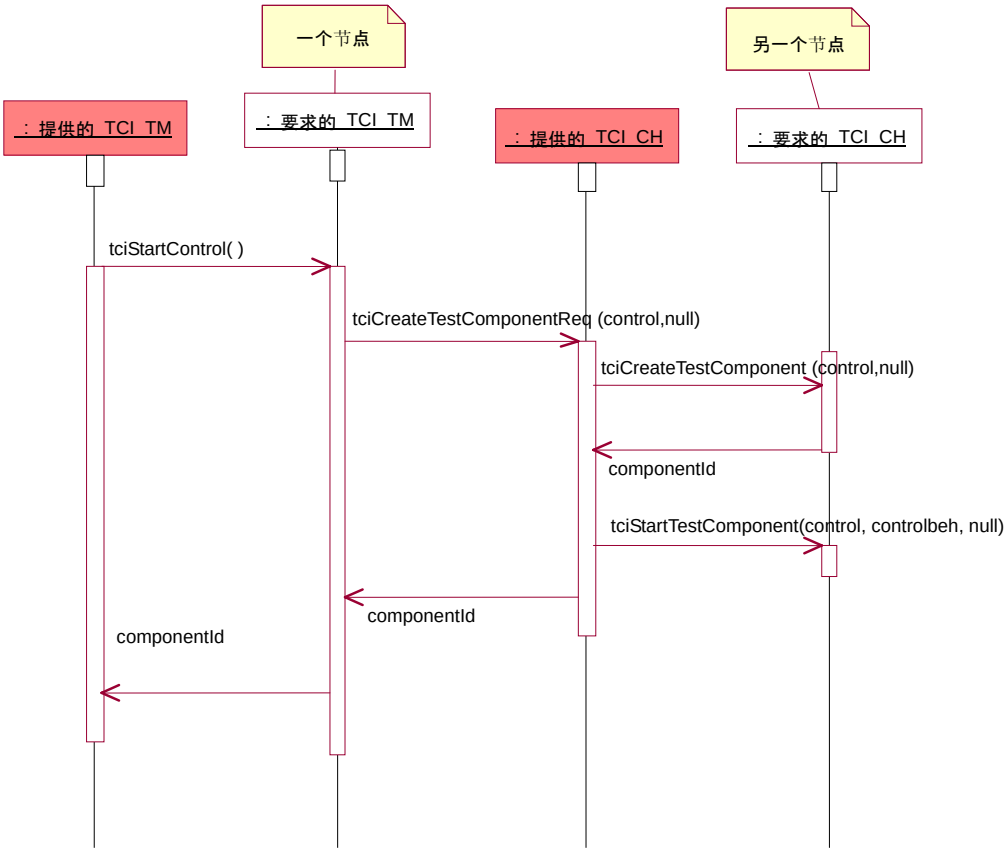

图 17/Z.145－使用方案 — 创建远程控制部件

11.3.2.2 TTCN-3段

```
module AModule {
    …
    control {
        …
    }
    …
}
```

11.3.3 使用方案：创建本地MTC

在图 18 中的方案显示了主测试部件的本地创建。本地意味着两种情况：

1） 在相同节点上，到测试管理TCI-TM的用户接口位于该节点上（当直接启动一个测试用例时）；或

2） 在相同控制部件位于的节点上（当从一个控制部分执行一个测试用例时）。

只要执行一个测试用例，就可以创建一个主测试部件：把创建测试部件请求发送到 TCI-CH，TCI-CH 再把请求发送到主测试部件应被创建的 TE 处。在这种情况下，TE 在相同的节点上。返回主测试部件的标识符并发送给 TCI-TM。然后使用标识符来启动主测试部件上的测试用例行为（在此没有示出，但是处理方法与第 11.3.5 节和第 11.3.6 节中描述的方案相同）。

11.3.3.1 序列图

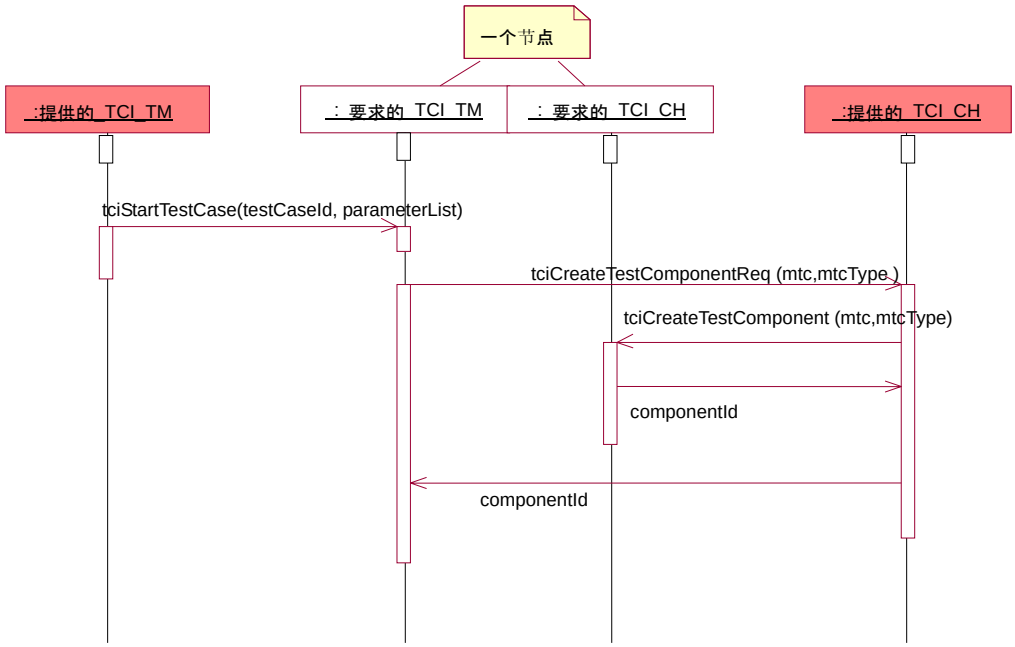

图 18/Z.145－使用方案 — 创建本地MTC

11.3.3.2 TTCN-3段

```
module AModule {
    …
    testcase ATestCase (…)runs on MTCType… {
        … //the test case behaviour
    }
    …
}
```

11.3.4 使用方案：创建远程MTC

在图 19 中的方案显示了主测试部件的远程创建。远程意味着两种情况：

1） 在另一个节点上，到测试管理TCI-TM的用户接口位于该节点上（当直接启动一个测试用例时）；或

2） 在控制部件位于的另一个节点上（当从一个控制部分执行一个测试用例时）。

只要执行一个测试用例，就可以创建一个主测试部件：把创建测试部件请求发送到 TCI-CH，TCI-CH 再把请求发送到主测试部件应被创建的 TE 处。在这种情况下，TE 在另一个节点上。返回主测试部件的标识符并发送给 TCI-TM。然后使用标识符来启动主测试部件上的测试用例行为（在此没有示出，但是处理方法与第 11.3.5 节和第 11.3.6 节中描述的方案相同）。

11.3.4.1 序列图

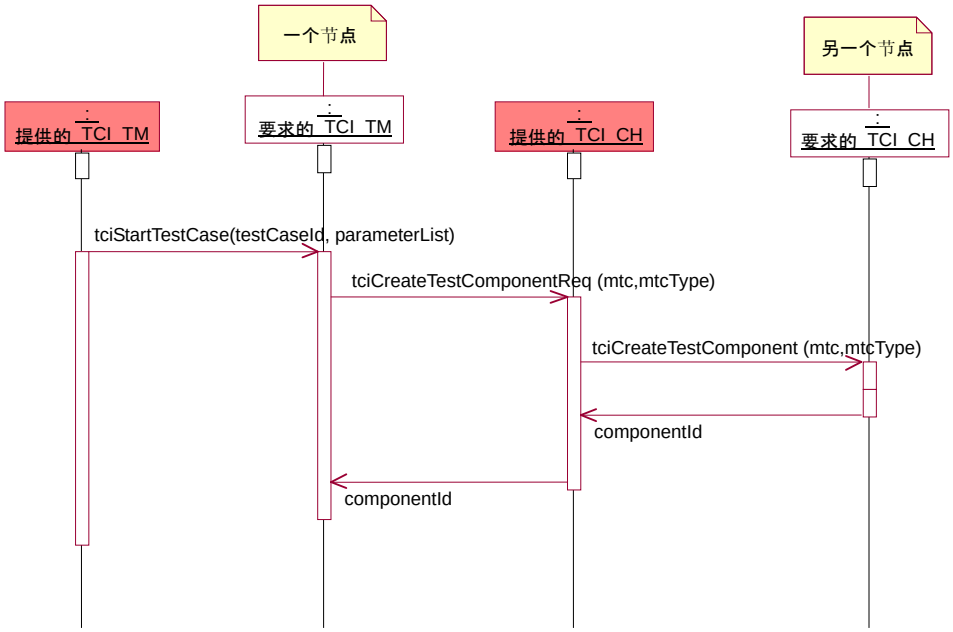

图 19/Z.145－使用方案 — 创建远程MTC

11.3.4.2 TTCN-3段

```
module AModule {
    …
    testcase ATestCase(…)runs on MTCType … {
        … //the test case behaviour
    }
    …
}
```

11.3.5 使用方案：在控制部分内执行测试用例的部件处理

在图 20 中的方案显示了在控制部分内执行测试用例的部件处理。当启动控制部分时，创建一个控制部件并把部件标识符返回给测试管理。对于在控制部分内要执行的每个测试用例，创建一个主测试部件并把部件标识符返回给控制部件。随后在主测试部件上启动测试用例行为，并通知测试管理关于测试用例的启动。当主测试部件终止时，对主测试部件终止和本地判定的请求一起被发送，使全球测试判定产生并得到关于测试用例终止的信息。

11.3.5.1 序列图

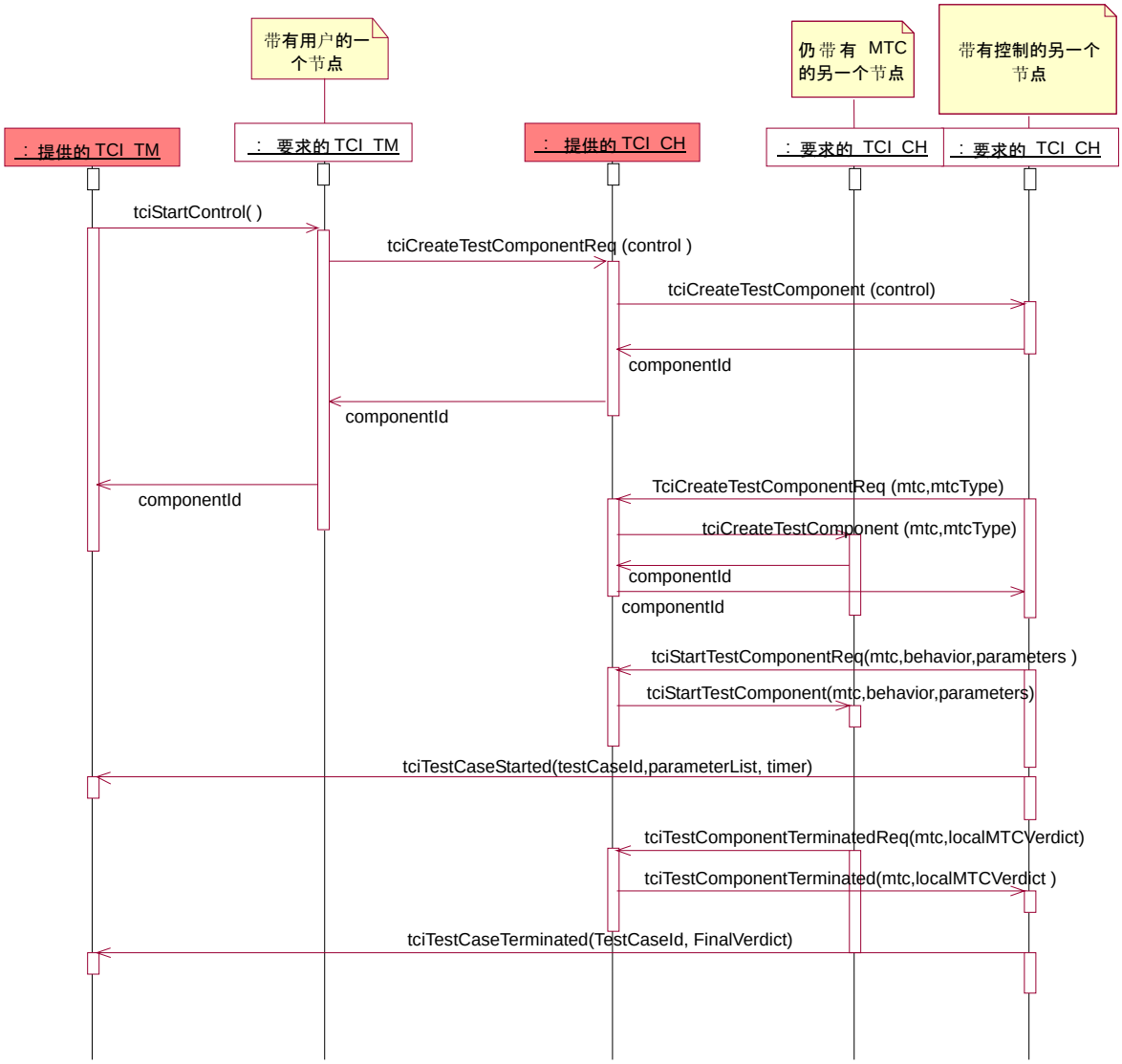

图 20/Z.145－使用方案 — 在控制部分内执行测试用例的部件处理

11.3.5.2 TTCN-3段

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3.6 使用方案：直接执行测试用例的部件处理

在图 21 中的方案显示了直接执行测试用例时，即在一个控制部分外执行时，如何处理测试部件。当启动测试用例时，创建主测试部件且首先在该主测试部件上启动测试用例行为。只要在一个测试用例内使用并行测试部件，则用相同的方式处理：首先启动并行测试部件：发送一个创建测试部件的请求到 TCI-CH 实体，该实体再把创建测试部件发送到 TE，在该 TE 内应创建并行测试部件。返回创建的并行测试部件的标识符。然后使用该标识符启动 PTC 行为用于启动操作。当 PTC 终止其执行时，一起发出测试部件终接请求和本地测试判定，以通知 TCI-CH 关于该终止。当主测试部件终止时，完成相同过程。另外，主测试部件的终止导致测试用例的完全终止。

11.3.6.1 序列图

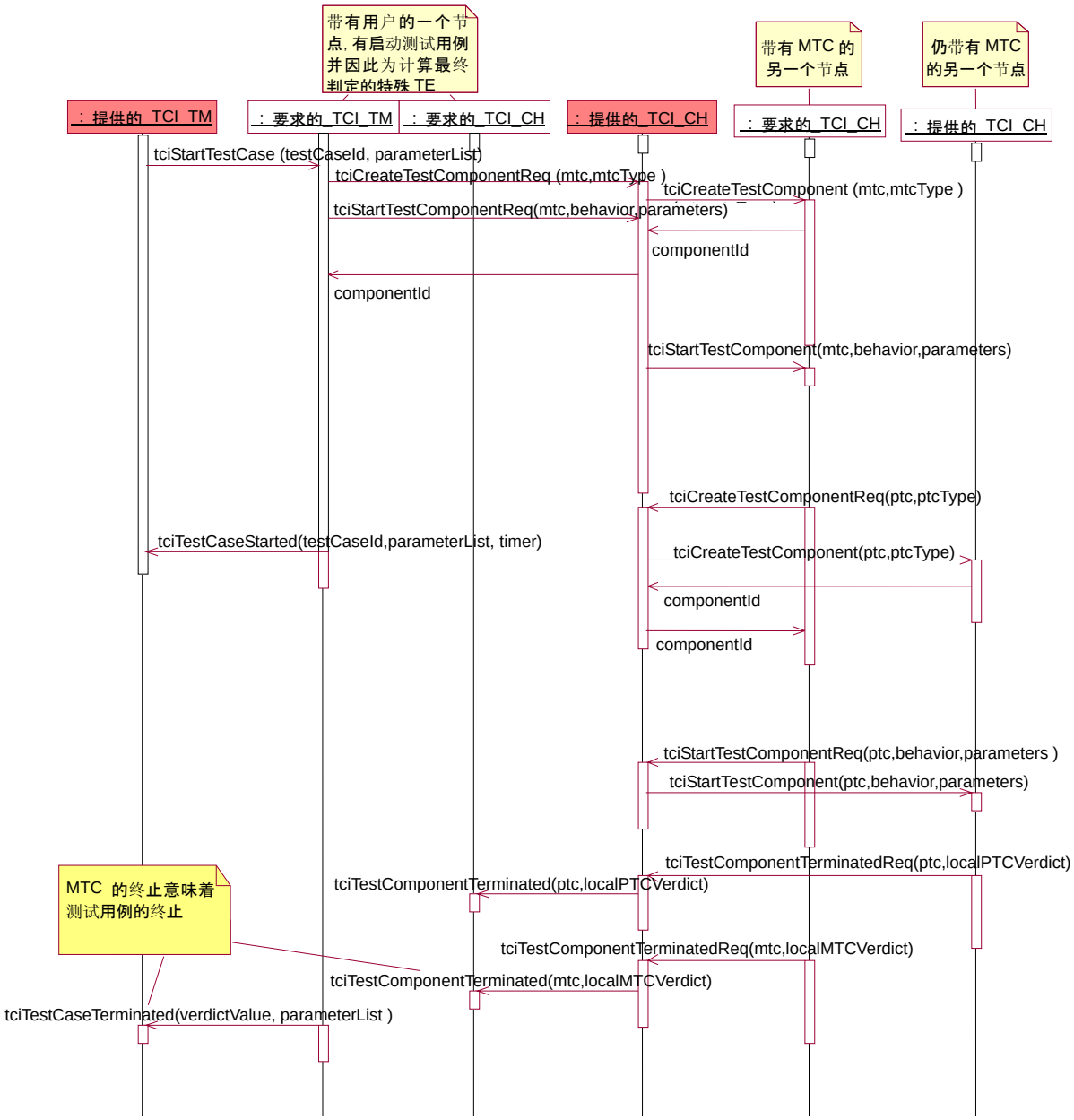

图 21/Z.145－使用方案 — 直接执行测试用例的部件处理

11.3.6.2 TTCN-3段

```
module AModule {
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    }
    …
    testcase ATestCase(…)… {
        … //the test case behaviour
        var APTCType PTC:= APTCType.create;
        …
        PTC.start(APTCBehaviour(…));
        …
    }
    …
}
```

11.3.7 使用方案：映射/连接的发送

在图 22 中的方案显示了如何映射端口。把映射一个端口的请求发送到 TE，最后在 TE 处执行映射。连接请求的发送也是类似的情况。

11.3.7.1 序列图

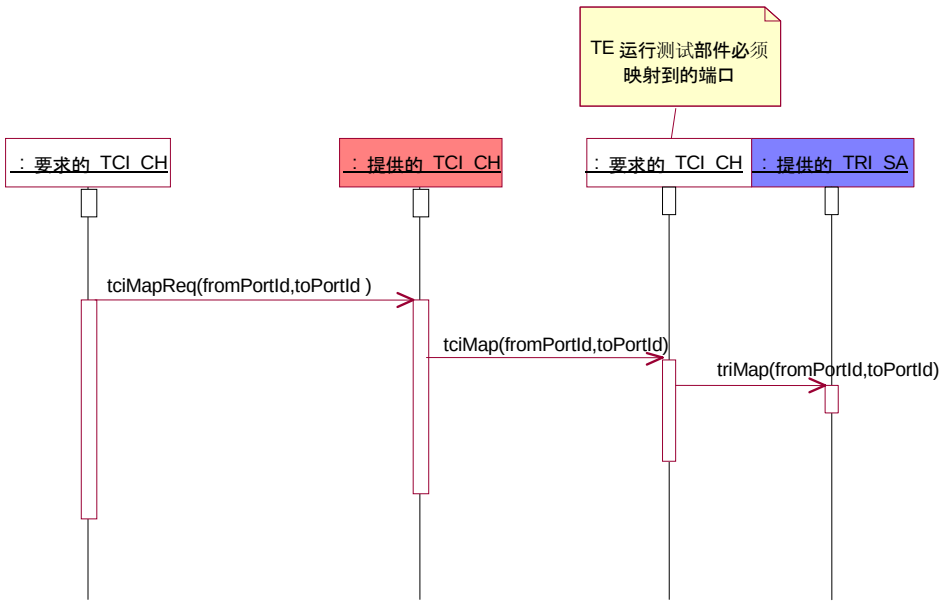

图 22/Z.145－使用方案 — 映射的发送

11.3.7.2 TTCN-3段

```
module AModule {
    …
    type port A { … }
    type component CA { port A a }
        type component CB { port A a }
    …
    testcase ATestCase(…)runs on CA system CB {
        var CA ptc := CA.create;
        … //the test case behaviour
        map(ptc:a,system:a);
        …
    }
    …
}
```

11.3.8 使用方案：取消映射/断开连接的发送

在图 23 中的方案显示了如何取消映射端口。把取消映射一个端口的请求发送到 TE，最后在 TE 处执行取消映射。断开连接请求的发送也是类似的情况。

11.3.8.1 序列图

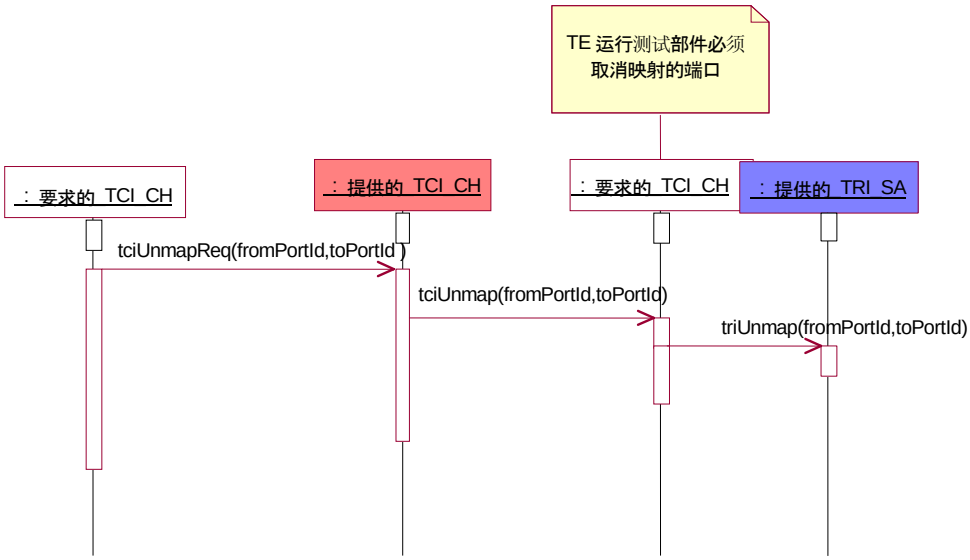

图 23/Z.145－使用方案 — 取消映射的发送

11.3.8.2 TTCN-3段

```
module AModule {
    …
    type port A { … }
    type component CA { port A a }
    type component CB { port A a }
    …
    testcase ATestCase(…)runs on CA system CB {
        var CA ptc := CA.create;
        … //the test case behaviour
        unmap(ptc:a,system:a);
        …
    }
    …
}
```

11.4 测试用例的终止和控制

11.4.1 使用方案：停止测试用例

在图 24 中的方案显示了在执行测试用例的过程中如何从测试管理停止测试用例。一旦 TM 收到关于一个启动测试用例的信息，则能够请求一个停止测试用例，直到接收到测试用例已经被终止的信息。一旦停止测试用例，所有并行测试部件应被停止且测试系统应被重置。

11.4.1.1 序列图

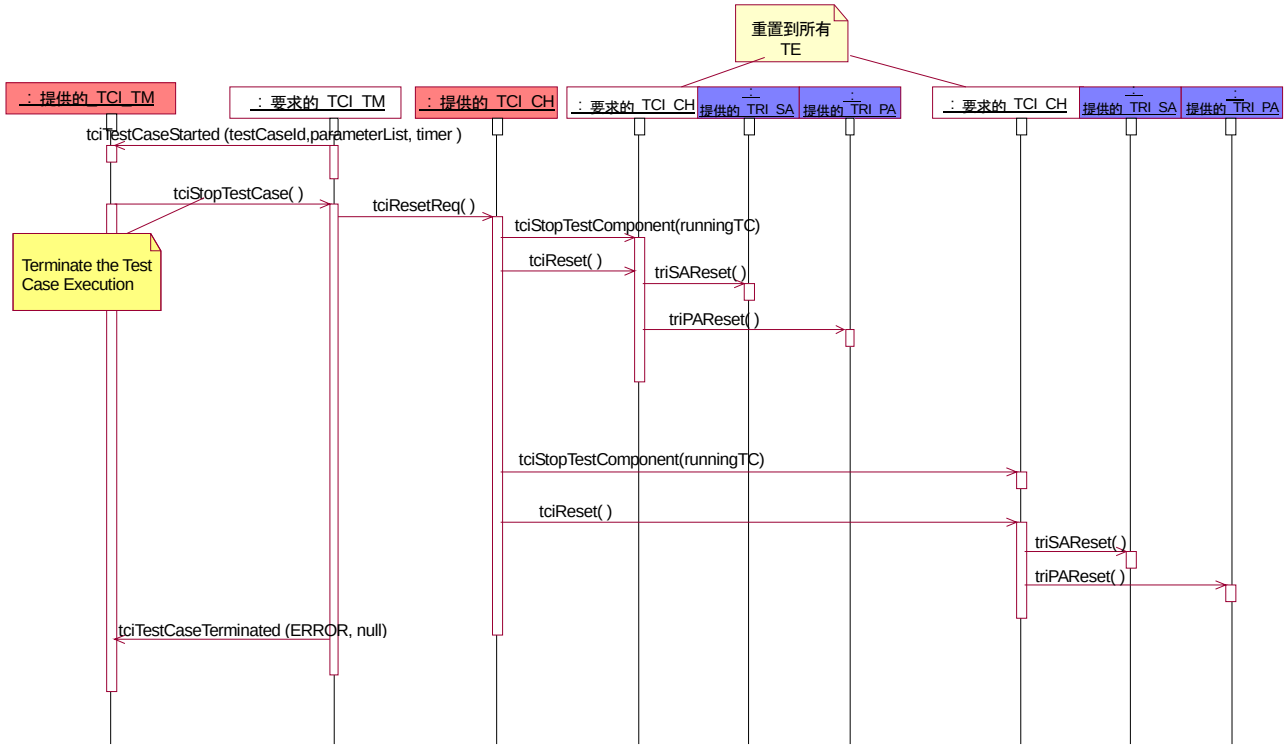

图 24/Z.145－使用方案 — 停止测试用例

11.4.1.2 TTCN-3段

没有 TTCN-3 码与 TM 如何选择执行测试用例的终止有关。这不在 TTCN-3 的范围内。

11.4.2 使用方案：停止控制

在图 25 中的方案显示了在执行控制部分的过程中如何从测试管理停止控制部分。可以在启动控制和终止之间停止控制部分。如果在测试用例正在执行时，控制部分接收到停止测试用例请求，则应停止正在执行的测试用例。而且，测试系统应被重置，正如图 24 中所描述的。

11.4.2.1 序列图

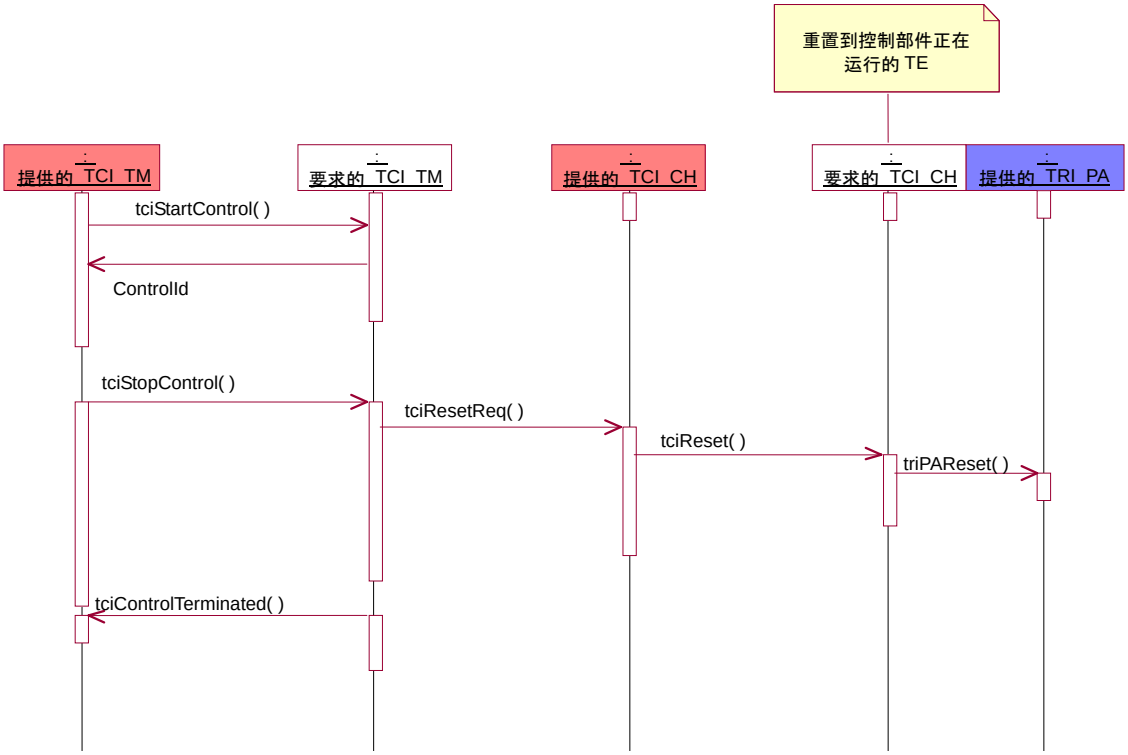

图 25/Z.145－使用方案 — 停止控制

11.4.2.2 TTCN-3段

从测试管理停止控制部分不在 TTCN-3 的范围内，因此不存在 TTCN-3 段。

11.4.3 使用方案：出错后控制的终止

在图 26 中的方案显示了当没有正在执行的测试用例时，在执行控制部分的过程中出错情况的处理。通知测试管理关于出错情况，然后必须明确终止控制部分的执行。一旦终止控制部分，测试系统应被重置。

11.4.3.1 序列图

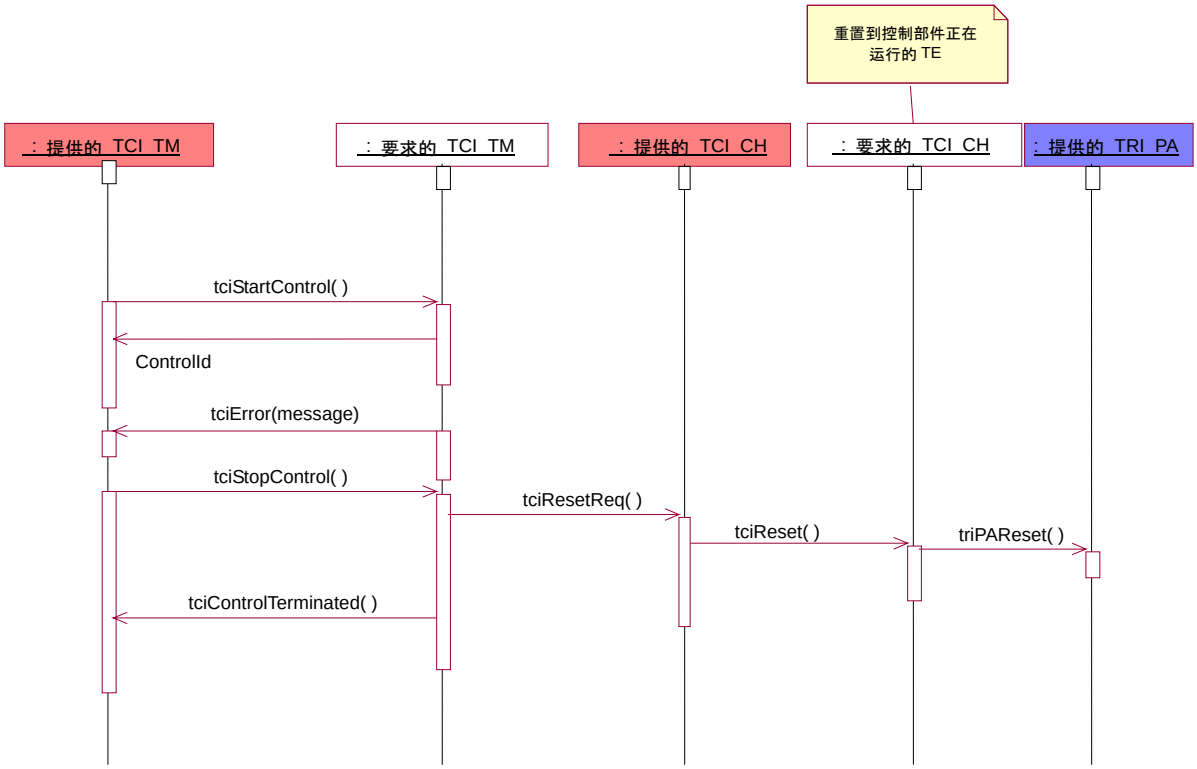

图 26/Z.145－使用方案 — 出错后控制的终止

11.4.3.2 TTCN-3段

由于在一个测试系统中，出错状况是例外情况，因此该方案没有 TTCN-3 段，也没有一个对应的 TTCN-3 的概念。相当于 TTCN-3 语义描述了在一个测试系统中，各种可能的出错状况。

11.4.4 使用方案：出错后测试用例的终止

在图 27 中的方案显示了直接执行测试用例的过程中出错情况的处理。通知测试管理关于出错情况，然后 TM 必须明确终止测试用例的执行。一旦停止测试用例，并行测试部件应被停止且测试系统应被重置。

11.4.4.1 序列图

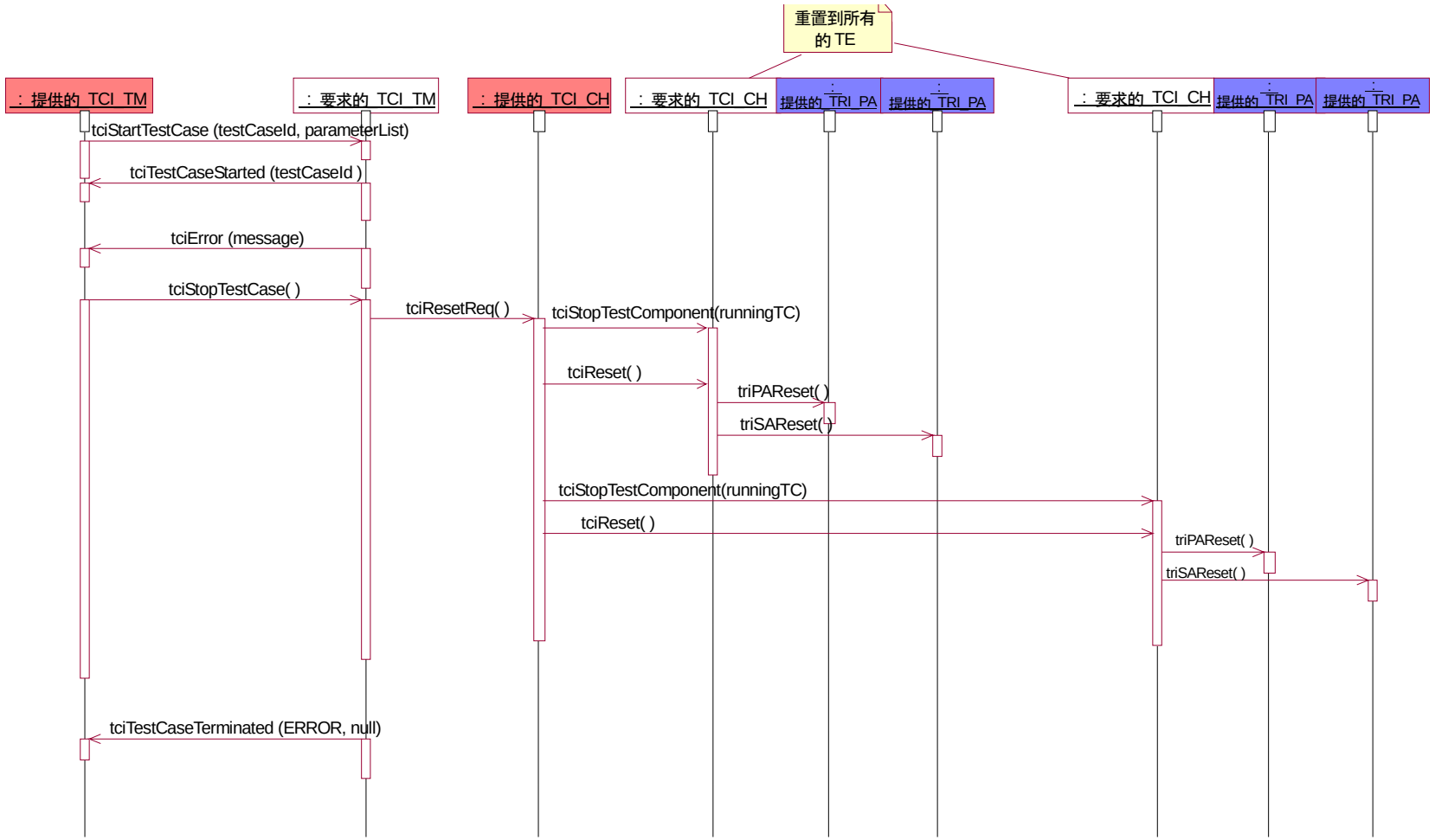

图 27/Z.145－使用方案 — 出错后测试用例的终止

11.4.4.2 TTCN-3段

由于在一个测试系统中，出错状况是例外情况，因此该方案没有 TTCN-3 段，也没有一个对应的 TTCN-3 的概念。相当于 TTCN-3 语义描述了在一个测试系统中，各种可能的出错状况。

11.4.5 使用方案：重置

在图 28 中的方案显示了测试系统的重置。在这种情况下，所有涉及的 TE 与他们的 TRI 系统适配器（SA）和平台适配器（PA）一起被重置。

11.4.5.1 序列图

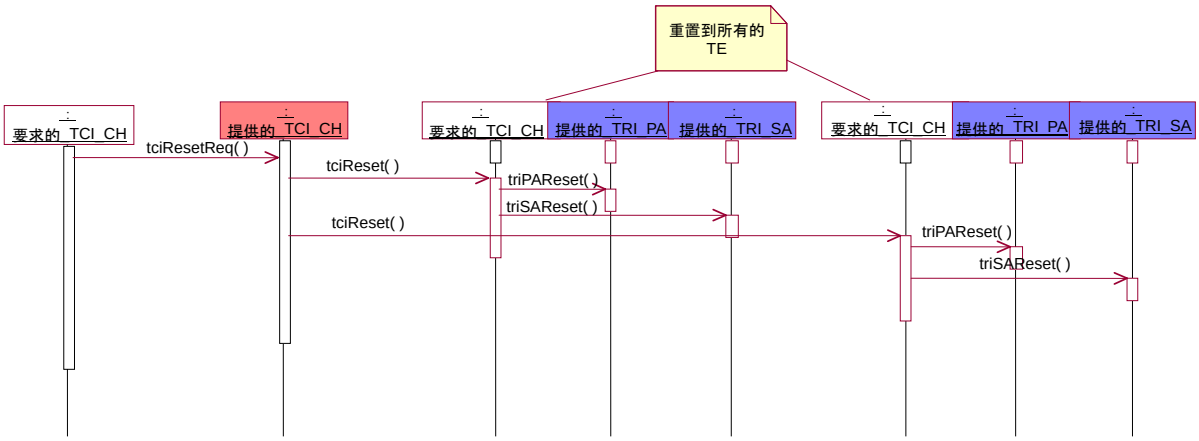

图 28/Z.145－使用方案 — 重置

11.4.5.2 TTCN-3段

由于在一个测试系统中，出错状况之后要求重置是例外情况，因此该方案没有 TTCN-3 段，也没有一个对应的 TTCN-3 的概念。

11.5 通信

11.5.1 使用方案：本地部件间通信

在图 29 中的方案显示了位于相同节点上的测试部件（主测试部件或并行测试部件）之间的通信。把通信请求发给 TCI-CH，然后 TCI-CH 决定在哪里排队等待通信模板。在这种情况下，通过在相同节点上的 TE 完成本地通信。本方案显示了使用发送操作基于消息的通信 — 该方案对于调用、应答和引起操作是相同的。

11.5.1.1 序列图

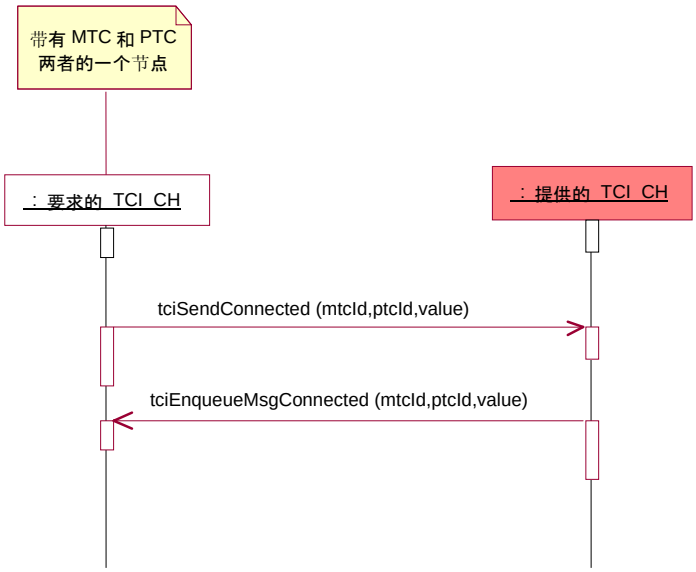

图 29/Z.145－使用方案 — 本地部件间通信

11.5.1.2 TTCN-3段

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //把数据发送到一个测试部件
        …
    }
    …
}
```

11.5.2 使用方案：测试部件间的节点间通信

在图 30 中的方案显示了位于不同节点上的测试部件（主测试部件或并行测试部件）之间的通信。把通信请求发给 TCI-CH，然后 TCI-CH 决定在哪里排队等待通信模板。在这种情况下，通过在另一个节点上的 TE 完成远程通信。本方案显示了使用发送操作基于消息的通信 — 该方案对于调用、应答和引起操作是相同的。

11.5.2.1 序列图

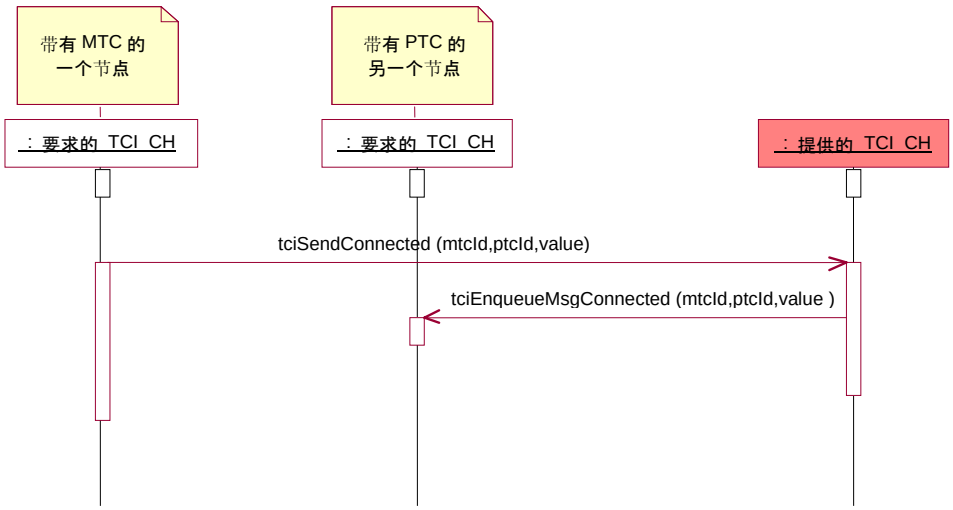

图 30/Z.145－使用方案 — 测试部件间的节点间通信

11.5.2.2 TTCN-3段

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //把数据发送到一个测试部件
        …
    }
    …
}
```

11.5.3 使用方案：编码

在图 31 中的方案显示了发送给 SUT 的数据的编码。从编码/解码实体通过 TCI-CD 接收已编码的数据。已编码的值通过 TRI-SA 发送给 SUT。该方案对于调用、应答和引起操作是相同的。

11.5.3.1 序列图

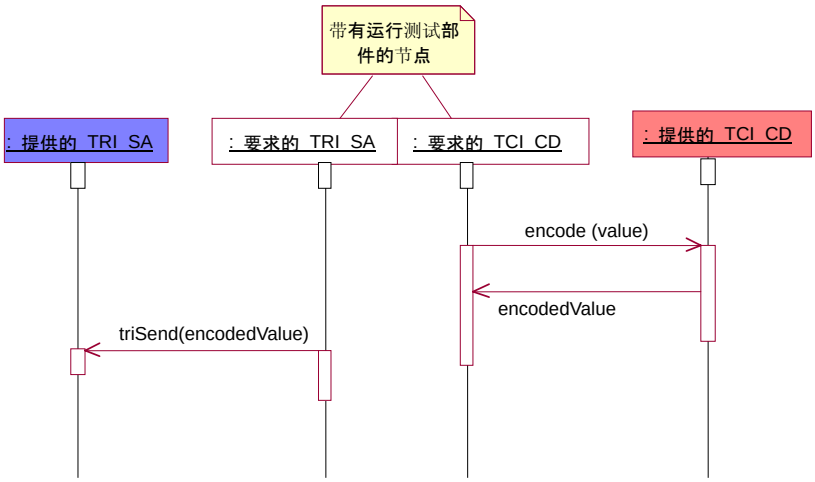

图 31/Z.145－使用方案 — 编码

11.5.3.2 TTCN-3段

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //the test case behaviour
        map(mtc:APort,system:APort);
        …
        APort.send(AMessageTemplate); //把数据发送到SUT
        …
    }
    …
} with { encoding = '…' }
```

11.5.4 使用方案：解码

在图 32 中的方案显示了从 SUT 通过 TRI-SA 接收到的数据的解码。从编码/解码实体通过 TCI-CD 接收已解码的数据。该方案对于接收、获得调用、获得应答、捕捉和检查操作是相同的。

11.5.4.1 序列图

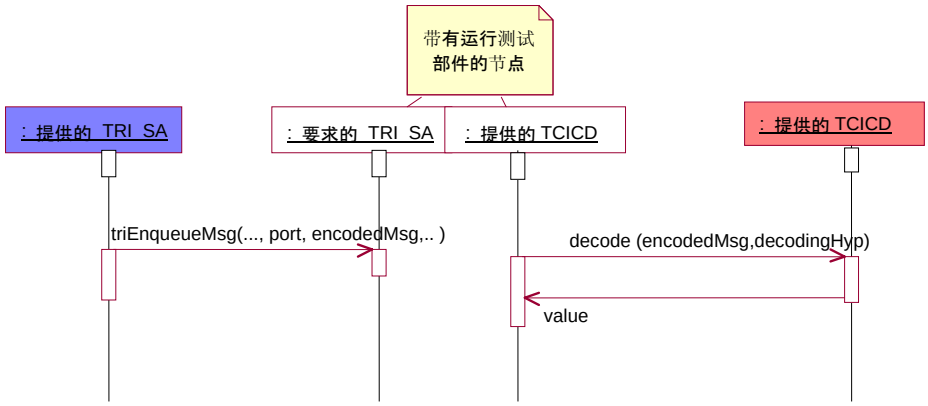

图 32/Z.145－使用方案 — 解码

11.5.4.2 TTCN-3段

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //the test case behaviour
        map(mtc:APort,system:APort);
        …
        APort.receive(AMessageTemplate); //从SUT接收数据
        …
    }
    …
} with { encoding = '…' }
```

附 件 A

TCI的IDL规范

本附件规定了使用接口定义语言（IDL）的 TTCN-3 控制接口。

```
// ********************************************************************************
// * 用于TTCN-3 控制接口的接口定义
// ********************************************************************************

module tciInterface {

  /* Forward declaration */
  interface Value;
  interface Type;

  // ********************************************************************************
  // * 取自TRI定义的数据类型
  // ********************************************************************************

  // 连接
  native TriPortIdType ;
  native TriPortIdListType;
  native TriComponentIdType ;
native TriComponentIdListType;

  // 通信
  native TriMessageType;
  native TriParameterType;
  native TriParameterListType;
  native TriAddressType;
  native TriAddressListType;
  native TriExceptionType;
  native TriSignatureIdType;

  // 其他
  native TriStatusType;
  native TriTimerIdType;
native TriTimerDurationType;

  // ********************************************************************************
  // * 一般抽象数据类型
  // ********************************************************************************

  // 基本定义
  native TBoolean;
  native TFloat;
  native TChar;
  native TInteger;
  native TString;
  native TUniversalChar;
  typedef sequence <TString> TStringSeq;
  native TObjid;

  struct QualifiedName {
    TString moduleName;
    TString baseName;
  };

  // 一般TCI抽象数据类型
  typedef QualifiedName TciBehaviourIdType;
  typedef QualifiedName TciModuleIdType;
  typedef QualifiedName TciModuleParameterIdType;
  typedef QualifiedName TciTestCaseIdType;

  enum TciParameterPassingModeType {
    IN_MODE,
    OUT_MODE,
    INOUT_MODE
  };

  struct TciParameterType {
    TciModuleParameterIdType parameterName;
    Value parameterValue;
    TciParameterPassingModeType mode;
  };
```

```
typedef sequence <TciParameterType> TciParameterListType;

struct TciParameterTypeType {
  Type parameterType;
  TciParameterPassingModeType mode;
};

typedef sequence <TciParameterTypeType> TciParameterTypeListType;

struct TciModuleParameterType {
  TciModuleParameterIdType parameterName;
  Value defaultValue;
};

typedef sequence <TciModuleIdType> TciModuleIdListType ;

typedef sequence <TciModuleParameterType> TciModuleParameterListType;

typedef sequence <TciTestCaseIdType> TciTestCaseIdListType;


enum TciTestComponentKindType {
  MTC,
  PTC,
  CONTROL
};

enum TciTypeClassType {
  ADDRESS_CLASS,
  ANYTYPE_CLASS,
  BITSTRING_CLASS,
  BOOLEAN_CLASS,
  CHAR_CLASS,
  CHARSTRING_CLASS,
  COMPONENT_CLASS,
  ENUMERATED_CLASS,
  FLOAT_CLASS,
  HEXSTRING_CLASS,
  INTEGER_CLASS,
  OBJID_CLASS,
  OCTETSTRING_CLASS,
  RECORD_CLASS,
  RECORDOF_CLASS,
  SET_CLASS,
  SETOF_CLASS,
  UNION_CLASS,
  UNIVERSALCHAR_CLASS,
  UNIVERSALCHARSTRING_CLASS,
  VERDICT_CLASS
};


// ***************************************************************************
// * 抽象TTCN-3数据类型和值
// ***************************************************************************

// 抽象数据类型"Type"
interface Type {
  TciModuleIdType  getDefiningModule ();
  TString          getName ();
  TciTypeClassType getTypeClass ();
  Value            newInstance ();
  TString          getTypeEncoding ();
  TString          getTypeEncodingVariant ();
  TStringSeq       getTypextension ();
};

// 抽象TTCN-3值
interface Value {
  TString  getValueEncoding ();
  TString  getValueEncodingVariant ();
  Type     getType ();
  TBoolean  notPresent ();
};

interface RecordOfValue : Value {
  Value    getField (in TInteger position);
  void     setField (
           in TInteger position,
```

```
            in Value value
              );
  void      appendField (in Value value);
  Type      getElementType ();
  TInteger getLength ();
  void      setLength (in TInteger len);
};

interface RecordValue : Value {
  Value       getField (in TString fieldName);
  void        setField (
            in TString fieldName,
            in Value value
            );
  TStringSeq getFieldNames ();
};

interface VerdictValue : Value {
  TInteger getVerdict ();
  void     setVerdict (in TInteger verdict);
};

interface BitstringValue : Value {
  TString   getString ();
  void       setString (in TString value);
  TInteger  getBit (in TInteger position);
  void       setBit (
            in TInteger position,
            in TInteger value
            );
  TInteger  getLength ();
  void       setLength (in TInteger len);
};

interface OctetstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TInteger getOctet (in TInteger position);
  void     setOctet (
            in TInteger position,
            in TInteger value
            );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface FloatValue : Value {
  TFloat getFloat ();
  void  setFloat (in TFloat value);
};

interface HexstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TInteger getHex (in TInteger position);
  void     setHex (
            in TInteger position,
            in TInteger value
            );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface ObjidValue : Value {
  TObjid getObjid ();
  void   setObjid (in TObjid value);
};

interface EnumeratedValue : Value {
  void    setEnum (in TString enumValue);
  TString getEnum ();
};

interface IntegerValue : Value {
  TInteger getInt ();
  void     setInt (in TInteger value);
};

interface CharValue : Value {
  TChar getChar ();
```

```
  void  setChar (in TChar value);
};

interface CharstringValue : Value {
  TString  getString ();
  void     setString (in TString value);
  TChar    getChar   (in TInteger position);
  void     setChar   (
           in TInteger position,
           in TChar value
           );
  TInteger getLength ();
  void     setLength (in TInteger len);
};

interface BooleanValue : Value {
  TBoolean getBoolean ();
  void    setBoolean (in TBoolean value);
};

interface UniversalCharValue : Value {
  TUniversalChar getUniversalChar ();
  void           setUniversalChar (in TUniversalChar value);

};

interface UniversalCharstringValue : Value {
  TString        getString ();
  void           setString (in TString value);
  TUniversalChar getChar   (in TInteger position);
  void           setChar   (
           in TInteger position,
           in TUniversalChar value
           );
  TInteger       getLength ();
  void           setLength (in TInteger len);
};

interface UnionValue : Value {
  Value      getVariant  (in TString variantName);
  void       setVariant  (
           in TString variantName,
           in Value value
           );
  TString    getPresentVariantName ();
  TStringSeq getVariantNames ();

};

// ********************************************************************************
// * 抽象记录类型
// ********************************************************************************

interface TciValueTemplate : Value {
    boolean isOmit ();
    boolean isAny();
    boolean isAnyOrOmit();
    TString getTemplateDef();
};

interface TciNonValueTemplate {
    boolean isAny();
    boolean isAll();
    TString getTemplateDef();
};

typedef sequence <Value> TciValueList;

struct TciValueDifference
{
  TString desc;
  Value val;
  TciValueTemplate tmpl;
};

typedef sequence <TciValueDifference> TciValueDifferenceList;
```

```
// ********************************************************************************
// 编解码接口
// -  要求的
// ********************************************************************************

interface TCI_CD_Required {
  Type getTypeForName (in TString typeName);
  Type getInteger ();
  Type getFloat ();
  Type getBoolean ();
  Type getChar ();
  Type getUniversalChar ();
  Type getObjid ();
  Type getCharstring ();
  Type getUniversalCharstring ();
  Type getHexstring ();
  Type getBitstring ();
  Type getOctetstring ();
  Type getVerdict ();
  void tciErrorReq (in TString message);
};

// ********************************************************************************
// 编解码接口
// -  提供的
// ********************************************************************************

interface TCI_CD_Provided {
  Value decode (
        in TriMessageType message,
        in Type decodingHypothesis
        );
  TriMessageType encode (in Value value);
};


// ********************************************************************************
// 测试管理接口
// -  要求的
// ********************************************************************************

interface TCI_TM_Required : TCI_CD_Required {
  void tciRootModule (in TciModuleIdType moduleName);
  TciModuleIdListType getImportedModules();
  TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName);
  TciTestCaseIdListType tciGetTestCases ();
  TciParameterTypeListType tciGetTestCaseParameters (
          in TciTestCaseIdType testCaseId
          );
  TriPortIdListType tciGetTestCaseTSI (
          in TciTestCaseIdType testCaseId
          );
  void tciStartTestCase (
          in TciTestCaseIdType testCaseId,
          in TciParameterListType parameterList
          );
  void tciStopTestCase ();
  TriComponentIdType tciStartControl ();
  void tciStopControl ();
};

// ********************************************************************************
// 测试管理接口
// -  提供的
// ********************************************************************************

interface TCI_TM_Provided {
  void tciTestCaseStarted (
          in TciTestCaseIdType testCaseId,
          in TciParameterListType parameterList,
          in TFloat timer
          );
  void tciTestCaseTerminated (
          in VerdictValue verdict,
          in TciParameterListType parameterList
          );
  void tciControlTerminated ();
  Value tciGetModulePar (
          in TciModuleParameterIdType parameterId
          );
```

```
  void tciLog (
       in TriComponentIdType testComponentId,
       in TString message
       );
  void tciError (in TString message);
};


// ********************************************************************************
// 部件处理接口
// -  要求的
// ********************************************************************************

interface TCI_CH_Required : TCI_CD_Required {
  void tciEnqueueMsgConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in Value receivedMessage
          );
  void tciEnqueueCallConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList
          );
  void tciEnqueueReplyConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList,
          in Value returnValue
          );
  void tciEnqueueRaiseConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in Value except
          );
  TriComponentIdType tciCreateTestComponent (
          in TciTestComponentKindType kind,
          in Type componentType,
          in String name
          );
  void tciStartTestComponent (
          in TriComponentIdType comp,
          in TciBehaviourIdType behavior,
          in TciParameterListType parameterList
          );
  void tciStopTestComponent (
          in TriComponentIdType comp
          );
  void tciConnect (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciDisconnect (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciTestComponentTerminated (
         in TriComponentIdType comp,
         in VerdictValue verdict
         );
  TBoolean tciTestComponentRunning (
         in TriComponentIdType comp
         );
  TriComponentIdType tciGetMTC ();
  void tciMap (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciUnmap (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciExecuteTestCase (
         in TciTestCaseIdType testCaseId,
         in TriPortIdListType tsiPortList
         );
```

```
  TBoolean tciTestComponentDone (
         in TriComponentIdType comp
         );
  void tciReset ();
};

// ********************************************************************************
// 部件处理接口
// -  提供的
// ********************************************************************************

interface TCI_CH_Provided {
  void tciSendConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in Value sendMessage
          );
  void tciSendConnectedBC (
         in TriPortIdType sender,
         in Value sendMessage
         );
  void tciSendConnectedMC (
         in TriPortIdType sender,
         in TriComponentIdListType receivers,
         in Value sendMessage
         );

  void tciCallConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList
          );
  void tciCallConnectedBC (
        in TriPortIdType sender,
        in TriSignatureIdType signature,
        in TciParameterListType parameterList
        );
  void tciCallConnectedMC (
        in TriPortIdType sender,
        in TriComponentIdListType receivers,
        in TriSignatureIdType signature,
        in TciParameterListType parameterList
        );

  void tciReplyConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList,
          in Value returnValue
          );
  void tciReplyConnectedBC (
         in TriPortIdType sender,
         in TriSignatureIdType signature,
         in TciParameterListType parameterList,
         in Value returnValue
         );
  void tciReplyConnectedMC (
         in TriPortIdType sender,
         in TriComponentIdListType receivers,
         in TriSignatureIdType signature,
         in TciParameterListType parameterList,
         in Value returnValue
         );

  void tciRaiseConnected (
          in TriPortIdType      sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in Value except
          );
  void tciRaiseConnectedBC (
        in TriPortIdType sender,
        in TriSignatureIdType signature,
        in Value except
        );
  void tciRaiseConnectedMC (
        in TriPortIdType sender,
        in TriComponentIdListType receivers,
```

```
        in TriSignatureIdType signature,
        in Value except
        );

  TriComponentIdType tciCreateTestComponentReq (
          in TciTestComponentKindType kind,
          in Type componentType,
          in String name
          );
  void tciStartTestComponentReq (
          in TriComponentIdType comp,
          in TciBehaviourIdType behavior,
          in TciParameterListType parameterList
          );
  void tciStopTestComponentReq (
          in TriComponentIdType comp
          );
  void tciConnectReq (
          in TriPortIdType fromPort,
          in TriPortIdType toPort
          );
  void tciDisconnectReq (
          in TriPortIdType fromPort,
          in TriPortIdType toPort
          );
  void tciTestComponentTerminatedReq (
          in TriComponentIdType comp,
          in VerdictValue verdict
          );
  TBoolean tciTestComponentRunningReq (
          in TriComponentIdType comp
          );
  TriComponentIdType tciGetMTCReq ();
  void tciMapReq (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciUnmapReq (
         in TriPortIdType fromPort,
         in TriPortIdType toPort
         );
  void tciExecuteTestCaseReq (
          in TciTestCaseIdType testCaseId,
          in TriPortIdListType tsiPortList
          );
  void tciResetReq ();
  TBoolean tciTestComponentDoneReq (
          in TriComponentIdType comp
          );
};

// *****************************************************************************
// 测试登录接口
// -  提供的
// *****************************************************************************

interface TCI_TL_Provided {
  void tliTcExecute(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciTestCaseIdType tcId,
         in TriParameterListType pars, in TriTimerDurationType dur
         );
  void tliTcStart(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciTestCaseIdType tcId,
         in TriParameterListType pars, in TriTimerDurationType dur
         );
  void tliTcStop(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c
         );
  void tliTcStarted(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciTestCaseIdType tcId,
         in TriParameterListType pars, in TriTimerDurationType dur
         );
  void tliTcTerminated(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciTestCaseIdType tcId,
         in TriParameterListType pars, in VerdictValue outcome);
```

```
void tliCtrlStart(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c
       );
void tliCtrlStop(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c
       );
void tliCtrlTerminated(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c);

void tliMSend_m(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriAddressType address, in TriStatusType encoderFailure,
       in TriMessageType msg, in TriStatusType transmissionFailure
       );
 void tliMSend_m_BC(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriStatusType encoderFailure, in TriMessageType msg,
       in TriStatusType transmissionFailure
       );
 void tliMSend_m_MC(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriAddressListType addresses, in TriStatusType encoderFailure,
       in TriMessageType msg, in TriStatusType transmissionFailure
       );

void tliMSend_c(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriComponentIdType to, in TriStatusType transmissionFailure
       );
 void tliMSend_c_BC(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriStatusType transmissionFailure
       );
 void tliMSend_c_MC(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriComponentIdListType toList, in TriStatusType transmissionFailure);

void tliMDetected_m(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in TriMessageType msg,
       in TriAddressType address
       );
void tliMDetected_c(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TriComponentIdType from
       );
void tliMMismatch_m(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs,
        in TriAddressType address, in TciValueTemplate addressTmpl
       );
void tliMMismatch_c(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs,
        in TriComponentIdType from, in TciNonValueTemplate fromTmpl
       );
void tliMReceive_m(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TciValueTemplate msgTmpl, in TriAddressType address,
       in TciValueTemplate addressTmpl
       );
void tliMReceive_c(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port, in Value msgValue,
       in TciValueTemplate msgTmpl, in TriComponentIdType from,
       in TciNonValueTemplate fromTmpl
```

```
        );
    void tliPrCall_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriAddressType address, in TriStatusType encoderFailure,
        in TriParameterListType pars, in TriStatusType transmissionFailure
        );
     void tliPrCall_m_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriStatusType encoderFailure, in TriParameterListType pars,
        in TriStatusType transmissionFailure
        );
     void tliPrCall_m_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriAddressListType addresses, in TriStatusType encoderFailure,
        in TriParameterListType pars,  in TriStatusType transmissionFailure
        );

    void tliPrCall_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriComponentIdType to, in TriStatusType transmissionFailure
        );
     void tliPrCall_c_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriStatusType transmissionFailure
        );
       void tliPrCall_c_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriComponentIdListType toList, in TriStatusType transmissionFailure
        );

    void tliPrGetCallDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TriParameterListType pars,
        in TriAddressType address
        );
    void tliPrGetCallDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TriComponentIdType from
        );
    void tliPrGetCallMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs,
        in TriAddressType address, in TciValueTemplate addressTmpl
        );
    void tliPrGetCallMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs,
        in TriComponentIdType from, in TciNonValueTemplate fromTmpl
        );
    void tliPrGetCall_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TciValueTemplate parsTmpl, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
    void tliPrGetCall_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
```

```
        in TciValueTemplate parsTmpl, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );

void tliPrReply_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TriAddressType address,
        in TriStatusType encoderFailure, in TriParameterType repl,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_m_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriStatusType encoderFailure, in TriParameterType repl,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_m_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriAddressListType addresses, in TriStatusType encoderFailure,
        in TriParameterType repl, in TriStatusType transmissionFailure
        );

void tliPrReply_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TriComponentIdType to,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_c_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_c_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriComponentIdListType toList, in TriStatusType transmissionFailure
        );

void tliPrGetReplyDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TriParameterType repl,
        in TriAddressType address
        );
void tliPrGetReplyDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value replValue,
        in TriComponentIdType from
        );
void tliPrGetReplyMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
        in TciValueDifferenceList diffs, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
void tliPrGetReplyMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
        in TciValueDifferenceList diffs, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );
void tliPrGetReply_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
```

```
        in TriAddressType address, in TciValueTemplate addressTmpl
        );
    void tliPrGetReply_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
        in TriComponentIdType from, in TciNonValueTemplate fromTmpl
        );

    void tliPrRaise_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TriAddressType address, in TriStatusType encoderFailure,
        in TriExceptionType exc, in TriStatusType transmissionFailure
        );
     void tliPrRaise_m_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc,
        in TriStatusType transmissionFailure
        );
     void tliPrRaise_m_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TriAddressListType addresses,
        in TriStatusType encoderFailure, in TriExceptionType exc,
        in TriStatusType transmissionFailure
        );

    void tliPrRaise_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TriComponentIdType to,
        in TriStatusType transmissionFailure
        );
    void tliPrCatchDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TriExceptionType exc,
        in TriAddressType address
        );
    void tliPrCatchDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value excValue,
        in TriComponentIdType from
        );
    void tliPrCatchMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
    void tliPrCatchMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );
    void tliPrCatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TriAddressType address, in TciValueTemplate addressTmpl);

    void tliPrCatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
```

```
       in TriComponentIdType from, in TciNonValueTemplate fromTmpl
       );
void tliPrCatchTimeoutDetected(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port,
       in TriSignatureIdType signature
       );
void tliPrCatchTimeout(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port,
       in TriSignatureIdType signature, in TciParameterListType parsValue
       );
void tliCCreate(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp,
       in String name
       );
void tliCStart(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp,
       in TciBehaviourIdType name, in TciParameterListType parsValue
       );
void tliCRunning(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status
       );
 void tliCAlive(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c,
         in TriComponentIdType comp, in TBoolean status
         );
 void tliCStop(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCKill(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCDoneMismatch(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCKilledMismatch(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCKilled(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
void tliCTerminated(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in VerdictValue verdict
       );
void tliPConnect(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPDisconnect(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPMap(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPUnmap(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPClear(
       in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
        in TriComponentIdType c, in TriPortIdType port
        );
    void tliPStart(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port
        );
    void tliPStop(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port
        );
    void tliPHalt(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port
        );
    void tliEncode(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in Value val, in TriStatusType encoderFailure,
        in TriMessageType msg, in TString codec
        );
    void tliDecode(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriMessageType msg,
        in TriStatusType decoderFailure, in Value val, in TString codec
        );
    void tliTTimeoutDetected(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer
        );
    void tliTTimeoutMismatch(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciNonValueTemplate timerTmpl
        );
    void tliTTimeout(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TciNonValueTemplate timerTmpl
        );
    void tliTStart(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer,
        in TriTimerDurationType dur
        );
    void tliTStop(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer
        );
    void tliTRead(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer,
        in TriTimerDurationType elapsed
        );
    void tliTRunning(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status
        );
    void tliSEnter(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString name, in TciParameterListType parsValue,
        in TString kind
        );
    void tliSLeave(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString name, in Value returnValue,
        in TString kind
        );
    void tliVar(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString name, in Value varValue
        );
    void tliModulePar(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TString name, in Value parValue
        );
    void tliGetVerdict(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in VerdictValue verdict
        );
    void tliSetVerdict(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in VerdictValue verdict
        );
```

```
    void tliLog(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c, in TciValueList log
           );
    void tliAEnter(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    void tliALeave(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    void tliADefaults(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    void tliAActivate(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c, in TString name, in TciParameterListType pars,
           in Value ref
           );
    void tliADeactivate(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c, in Value ref
           );
    void tliANomatch(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    void tliARepeat(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    void tliAWait(
           in TString am, in TInteger ts, in TString src, in TInteger line,
           in TriComponentIdType c
           );
    };
};
```

附 件 B

用于提供的TCI TL的XML映射

本附件规定使用扩展标记语言（XML）方案定义的用于 TCI 登录接口的映射。

B.1 用于简单类型的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
            xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
            elementFormDefault="qualified">

    <!-- 基本定义 -->
    <xsd:simpleType name="xpath">
        <!-- this string should be XPATH compliant -->
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TBoolean">
        <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>

    <xsd:simpleType name="TString">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TInteger">
        <xsd:restriction base="xsd:integer"/>
    </xsd:simpleType>

    <!-- 其他 -->
    <xsd:simpleType name="TriTimerDurationType">
        <xsd:restriction base="xsd:float"/>
    </xsd:simpleType>

    <xsd:simpleType name="TciParameterPassingModeType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="in"/>
            <xsd:enumeration value="inout"/>
            <xsd:enumeration value="out"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TriStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TRI_Ok"/>
            <xsd:enumeration value="TRI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TciStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TCI_Ok"/>
            <xsd:enumeration value="TCI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

B.2 用于类型的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
        elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
```

```
 schemaLocation="SimpleTypes.xsd"/>

<!—连接 -->
<xsd:complexType name="TriPortIdType">
    <xsd:sequence>
        <xsd:element name="comp" type="Types:TriComponentIdType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="port" type="Types:Port"  minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Port">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="index" type="xsd:int" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TriComponentIdType">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="null"/>
            <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>

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

<!—通信 -->
<xsd:complexType name="TriMessageType">
    <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>

<xsd:complexType name="TriParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="Values:Value" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="SimpleTypes:TString"/>
    <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
</xsd:complexType>

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

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

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

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

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


<!-- 其他 -->
<xsd:complexType name="TriTimerIdType">
    <xsd:sequence>
        <xsd:element name="id" type="Types:Id" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

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

    <!-- 基本定义 -->
    <xsd:complexType name="QualifiedName">
        <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- 一般TCI抽象数据类型 -->
    <xsd:complexType name="TciBehaviourIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TciTestCaseIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TciParameterType">
        <xsd:sequence>
            <xsd:element name="val" type="Values:Value" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="SimpleTypes:TString"/>
        <xsd:attribute name="mode" type="SimpleTypes:TciParameterPassingModeType"/>
    </xsd:complexType>

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

    <!-- 用于测试部件、端口和定时器的一般标识符结构 -->
    <xsd:complexType name="Id">
        <xsd:sequence>
           <xsd:element name="name" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="id" type="SimpleTypes:TInteger" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="type" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

B.3 用于值的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates.xsd"
     schemaLocation="Templates.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
     schemaLocation="SimpleTypes.xsd"/>

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

    <xsd:complexType name="Value" mixed="true">
        <xsd:choice>
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
```

```
        <xsd:element name="bitstring" type="Values:BitstringValue"/>
        <xsd:element name="hexstring" type="Values:HexstringValue"/>
        <xsd:element name="octetstring" type="Values:OctetstringValue"/>
        <xsd:element name="charstring" type="Values:CharstringValue"/>
        <xsd:element name="universal_charstring" type="Values:UniversalCharstringValue"/>
        <xsd:element name="record" type="Values:RecordValue"/>
        <xsd:element name="record_of" type="Values:RecordOfValue"/>
        <xsd:element name="set" type="Values:SetValue"/>
        <xsd:element name="set_of" type="Values:SetOfValue"/>
        <xsd:element name="enumerated" type="Values:EnumeratedValue"/>
        <xsd:element name="union" type="Values:UnionValue"/>
        <xsd:element name="anytype" type="Values:AnytypeValue"/>
        <xsd:element name="address" type="Values:AddressValue"/>
            </xsd:choice>
        <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<!-- 一般事件单元 -->
<xsd:complexType name="IntegerValue">
    <xsd:simpleContent>
        <xsd:extension base="SimpleTypes:TString">
                <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

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

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

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

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

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

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

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

```
</xsd:complexType>

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

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

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

<xsd:complexType name="RecordOfValue">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
```

```
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

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

<xsd:complexType name="SetOfValue">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
```

```
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="objid" type="Values:ObjidValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="bitstring" type="Values:BitstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="hexstring" type="Values:HexstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="octetstring" type="Values:OctetstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="charstring" type="Values:CharstringValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="universal_charstring"
                type="Values:UniversalCharstringValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record" type="Values:RecordValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="record_of" type="Values:RecordOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="set" type="Values:SetValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="set_of" type="Values:SetOfValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="enumerated" type="Values:EnumeratedValue"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="union" type="Values:UnionValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="anytype" type="Values:AnytypeValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="address" type="Values:AddressValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:choice>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

<xsd:complexType name="EnumeratedValue">
    <xsd:sequence>
        <xsd:element name="element" type="SimpleTypes:TString"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="Values:ValueAtts"/>
</xsd:complexType>

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

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

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

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

B.4 用于模板的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
     schemaLocation="SimpleTypes.xsd"/>
```

```
    <xsd:complexType name="TciValueTemplate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Values:Value">
            <xsd:choice>
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                <xsd:element name="float" type="Templates:FloatTemplate"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                <xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
            <xsd:element name="union" type="Templates:UnionTemplate"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
            <xsd:element name="address" type="Templates:AddressTemplate"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
             </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

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

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

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


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

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


    <xsd:complexType name="TciValueDifference">
            <xsd:attribute name="desc" type="SimpleTypes:TString" use="optional"/>
            <xsd:attribute name="val" type="SimpleTypes:xpath" use="required"/>
            <xsd:attribute name="tmpl" type="SimpleTypes:xpath" use="required"/>
```

```
    </xsd:complexType>

    <xsd:complexType name="TciValueDifferenceList">
            <xsd:sequence>
                    <xsd:element name="diff" type="Templates:TciValueDifference" minOccurs="1"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>


    <xsd:complexType name="IntegerTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:IntegerValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="FloatTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:FloatValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="BooleanTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:BooleanValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ObjidTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:ObjidValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="BitstringTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:BitstringValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
<xsd:complexType name="HexstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:BitstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="OctetstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:OctetstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CharstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:CharstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UniversalCharstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:UniversalCharstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RecordTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:RecordValue">
            <xsd:sequence>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
```

```
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    </xsd:choice>
                </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RecordOfTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:RecordOfValue">
            <xsd:choice>
                <xsd:sequence>
                    <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
```

```
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="SetTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:SetValue">
                <xsd:sequence>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                        <xsd:element name="float" type="Templates:FloatTemplate"/>
                        <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                        <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                        <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                        <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                        <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                        <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                        <xsd:element name="universal_charstring"
                            type="Templates:UniversalCharstringTemplate"/>
                    <xsd:element name="record" type="Templates:RecordTemplate"/>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                    <xsd:element name="set" type="Templates:SetTemplate"/>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                    <xsd:element name="union" type="Templates:UnionTemplate"/>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                    <xsd:element name="address" type="Templates:AddressTemplate"/>
                            <xsd:element name="omit" type="Templates:omit"/>
                            <xsd:element name="any" type="Templates:any"/>
                            <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="SetOfTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:SetOfValue">
                <xsd:choice>
                    <xsd:sequence>
                        <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                            minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                            minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                            minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:sequence>
                        <xsd:element name="charstring" type="Templates:CharstringTemplate"
                            minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
```

```
                <xsd:sequence>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="EnumeratedTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:EnumeratedValue">
            <xsd:choice>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UnionTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:UnionValue">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                <xsd:element name="float" type="Templates:FloatTemplate"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                <xsd:element name="universal_charstring"
                    type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
            <xsd:element name="union" type="Templates:UnionTemplate"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
```

```
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AnytypeTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AddressTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

B.5 用于事件的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events"
    xmlns:Events="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
```

```
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values" elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
     schemaLocation="SimpleTypes.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates.xsd"
     schemaLocation="Templates.xsd"/>

    <!-- 用于所有事件的通常定义 -->
    <xsd:complexType name="Event" mixed="true">
        <xsd:sequence>
            <xsd:element name="am" type="SimpleTypes:TString"/>
        </xsd:sequence>
        <xsd:attribute name="ts" type="xsd:time" use="required"/>
        <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

        <!-- 用于测试部件、端口和定时器的一般标识符结构 -->
        <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="id" type="SimpleTypes:TInteger" use="required"/>
        <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- 由所有端口配置事件扩展该事件 -->
    <xsd:complexType name="PortConfiguration">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port1" type="Types:TriPortIdType" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="port2" type="Types:TriPortIdType" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- 由所有状态事件扩展该事件 -->
    <xsd:complexType name="PortStatus">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- 测试用例 -->
    <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="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types: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="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

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

```
    </xsd:complexType>

    <xsd:complexType name="tliTcStarted">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types: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="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="outcome" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- 控制 -->
    <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>

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

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

    <xsd:complexType name="tliMSend_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/>
                    <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="port" type="Types:TriPortIdType"/>
                    <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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/>
                    <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="port" type="Types:TriPortIdType"/>
                    <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="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Types:TriMessageType"/>
                        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    </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="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Values:Value"/>
                        <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

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

    <xsd:complexType name="tliMReceive_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" 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>

     <!-- 同步通信 -->
    <xsd:complexType name="tliPrCall_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
```

```
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcallDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
```

```
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
```

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

    <xsd:complexType name="tliPrReply_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="from" type="Types:TriComponentIdType"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Values:Value"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Values:Value"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="from" type="Types:TriComponentIdType"/>
                    <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="replTmpl" type="Values:Value"/>
                    <xsd:element name="from" type="Types:TriComponentIdType"/>
                    <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
```

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

<xsd:complexType name="tliPrRaise_m_BC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                    <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="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                    <xsd:sequence>
                        <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                        <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="to" type="Types:TriComponentIdType" 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="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
    <xsd:complexType name="tliPrRaise_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="exc" type="Types:TriExceptionType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCatchDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="exc" type="Types:TriExceptionType"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" 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="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="exc" type="Values:Value"/>
                    <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCatchMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="exc" type="Values:Value"/>
                    <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                    <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
                </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="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="exception" type="Values:Value"/>
                <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/>
            </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="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="exception" type="Values:Value"/>
                <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="from" type="Types:TriComponentIdType"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
            </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="port" 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="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 部件 -->
<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: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="pars" 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:TBoolean"/>
            </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:TBoolean"/>
            </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:attribute name="done" type="SimpleTypes:TBoolean"/>
        </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:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDone">
    <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:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

 <xsd:complexType name="tliCKilled">
    <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:attribute name="done" type="SimpleTypes:TBoolean"/>
```

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

<xsd:complexType name="tliCTerminated">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="verdict" type="Values:VerdictValue" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 端口 -->
<xsd:complexType name="tliPConnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPDisconnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPMap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPUnmap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPClear">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStart">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStop">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPHalt">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<!-- 编解码 -->
<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"/>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
    <xsd:complexType name="tliDecode" mixed="true">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:choice>
                        <xsd:element name="val" type="Values:Value"/>
                        <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"/>
                    </xsd:choice>
                    <xsd:element name="msg" type="Types:TriMessageType"/>
                </xsd:sequence>
                <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- 定时器 -->
    <xsd:complexType name="tliTTimeoutDetected">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1" minOccurs="1"/>
                </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" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1"
minOccurs="1"/>
                </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" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1"
minOccurs="1"/>
                </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="Types: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: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="Types:TriTimerDurationType"/>
```

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

<xsd:complexType name="tliTRunning">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="timer" type="Types:TriTimerIdType"/>
            </xsd:sequence>
            <xsd:attribute name="status" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 范围 -->
<xsd:complexType name="tliSEnter">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="pars" type="Types:TriParameterListType" 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" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="return" type="Values:Value" minOccurs="0"/>
                <xsd:element name="kind" type="SimpleTypes:TString"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 变量和模块参数 -->
<xsd:complexType name="tliVar">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="val" type="Values:Value" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliModuleParr">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="val" type="Values:Value" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 判定 -->
<xsd:complexType name="tliGetVerdict">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="verdict" type="Values:VerdictValue"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliSetVerdict">
```

```
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="verdict" type="Values:VerdictValue"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- 记录 -->
<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>

<!-- 备选 -->
<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" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="pars" type="Types:TriParameterListType" 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:schema>
```

B.6 用于日志的TCI-TL XML方案

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI"
    xmlns:TLI="http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Events="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events" elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events.xsd"
     schemaLocation="Events.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"/>
        </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:time"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Trailer">
        <xsd:sequence/>
    </xsd:complexType>

    <xsd:complexType name="Body">
        <xsd:choice maxOccurs="unbounded">

            <!-- 测试用例操作 -->
            <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"/>

            <!-- 控制操作 -->
            <xsd:element name="tliCtrlStart" type="Events:tliCtrlStart"/>
            <xsd:element name="tliCtrlStop" type="Events:tliCtrlStop"/>
            <xsd:element name="tliCtrlTerminated" type="Events:tliCtrlTerminated"/>

            <!-- 异步通信 -->
            <xsd:element name="tliMSend_m" type="Events:tliMSend_m"/>
            <xsd:element name="tliMSend_c" type="Events:tliMSend_c"/>
            <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"/>

            <!-- 同步通信 -->
            <xsd:element name="tliPrCall_m" type="Events:tliPrCall_m"/>
            <xsd:element name="tliPrCall_c" type="Events:tliPrCall_c"/>

            <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="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="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="tliPrCatchTimeout"    type="Events:tliPrCatchTimeout"/>

            <!-- 部件 -->
            <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:tliCRunning"/>
            <xsd:element name="tliCStop" type="Events:tliCStop"/>
            <xsd:element name="tliCKill" type="Events:tliCStop"/>
            <xsd:element name="tliCDoneMismatch" type="Events:tliCDone"/>
            <xsd:element name="tliCDone" type="Events:tliCDone"/>
            <xsd:element name="tliCKilledMismatch" type="Events:tliCDone"/>
            <xsd:element name="tliCKilled" type="Events:tliCDone"/>
            <xsd:element name="tliCTerminated" type="Events:tliCTerminated"/>

            <!-- 端口 -->
            <xsd:element name="tliPConnect" type="Events:tliPConnect"/>
            <xsd:element name="tliPDisconnect" type="Events:tliPDisconnect"/>
            <xsd:element name="tliPMap" type="Events:tliPMap"/>
            <xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>
            <xsd:element name="tliPClear" type="Events:tliPClear"/>
            <xsd:element name="tliPStart" type="Events:tliPStart"/>
            <xsd:element name="tliPStop" type="Events:tliPStop"/>
            <xsd:element name="tliPHalt" type="Events:tliPStop"/>

            <!-- 编解码 -->
            <xsd:element name="tliDecode" type="Events:tliDecode"/>
            <xsd:element name="tliEncode" type="Events:tliEncode"/>

            <!-- 定时器 -->
            <xsd:element name="tliTTimeoutDetected" type="Events:tliTTimeoutDetected"/>
            <xsd:element name="tliTTimeoutMimatch" 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"/>

            <!-- 范围 -->
            <xsd:element name="tliSEnter" type="Events:tliSEnter"/>
            <xsd:element name="tliSLeave" type="Events:tliSLeave"/>

            <!-- 声明 -->
            <xsd:element name="tliVar" type="Events:tliVar"/>
            <xsd:element name="tliGetVerdict" type="Events:tliGetVerdict"/>
            <xsd:element name="tliSetVerdict" type="Events:tliSetVerdict"/>
            <xsd:element name="tliLog" type="Events:tliLog"/>

            <!-- 备选  -->
            <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"/>

        </xsd:choice>
    </xsd:complexType>
</xsd:schema>
```

参 考 资 料

- INTOOL CGI/NPL038 (V2.2): *Generic Compiler/Interpreter interface; GCI Interface Specification Infrastructural Tools for Informational Technology and Telcomunications Conference Testing*, December 1996.
- ITU-T Recommendation X.292 (2002), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – The Tree and Tabular Combined Notation* (*TTCN*).

  ISO/IEC 9646-3:1998, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3*: *The Tree and Tabular combined Notation* (*TTCN*).
- ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set* (*UCS*).
- OMG CORBA v2.2: *The Common Object Request Broker*: *Architecture and Specification*, Section 3, February 1998.

ITU-T 系列建议书

A系列	ITU-T工作的组织
D系列	一般资费原则
E系列	综合网络运行、电话业务、业务运行和人为因素
F系列	非话电信业务
G系列	传输系统和媒质、数字系统和网络
H系列	视听及多媒体系统
I系列	综合业务数字网
J系列	有线网络和电视、声音节目及其它多媒体信号的传输
K系列	干扰的防护
L系列	电缆和外部设备其它组件的结构、安装和保护
M系列	电信管理，包括TMN和网络维护
N系列	维护：国际声音节目和电视传输电路
O系列	测量设备的技术规范
P系列	电话传输质量、电话设施及本地线路网络
Q系列	交换和信令
R系列	电报传输
S系列	电报业务终端设备
T系列	远程信息处理业务的终端设备
U系列	电报交换
V系列	电话网上的数据通信
X系列	数据网、开放系统通信和安全性
Y系列	全球信息基础设施、互联网协议问题和下一代网络
Z系列	**用于电信系统的语言和一般软件问题**

瑞士印刷
2006年，日内瓦